

AirInv

0.1.2

Generated by Doxygen 1.8.0

Mon May 7 2012 08:37:23

Contents

1	AirInv Documentation	1
1.1	Getting Started	1
1.2	AirInv at SourceForge	1
1.3	AirInv Development	1
1.4	External Libraries	2
1.5	Support AirInv	2
1.6	About AirInv	2
2	People	2
2.1	Project Admins	2
2.2	Developers	2
2.3	Retired Developers	2
2.4	Contributors	3
2.5	Distribution Maintainers	3
3	Coding Rules	3
3.1	Default Naming Rules for Variables	3
3.2	Default Naming Rules for Functions	3
3.3	Default Naming Rules for Classes and Structures	3
3.4	Default Naming Rules for Files	3
3.5	Default Functionality of Classes	4
4	Copyright and License	4
4.1	GNU LESSER GENERAL PUBLIC LICENSE	4
4.1.1	Version 2.1, February 1999	4
4.2	Preamble	4
4.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	5
4.3.1	NO WARRANTY	9
4.3.2	END OF TERMS AND CONDITIONS	9
4.4	How to Apply These Terms to Your New Programs	9
5	Documentation Rules	10
5.1	General Rules	10
5.2	File Header	11
5.3	Grouping Various Parts	11
6	Main features	12
6.1	Network generation	12
6.2	Inventory generation	12
6.3	Finding travel solutions	12

6.4	Distributed inventories	12
6.5	Other features	12
7	Make a Difference	12
8	Make a new release	13
8.1	Introduction	13
8.2	Initialisation	13
8.3	Branch creation	13
8.4	Commit and publish the release branch	13
8.5	Update the change-log in the trunk as well	13
8.6	Create distribution packages	14
8.7	Generation the RPM packages	14
8.8	Update distributed change log	14
8.9	Create the binary package, including the documentation	14
8.10	Upload the files to SourceForge	14
8.11	Upload the documentation to SourceForge	15
8.12	Make a new post	15
8.13	Send an email on the announcement mailing-list	15
9	Installation	15
9.1	Table of Contents	15
9.2	Fedora/RedHat Linux distributions	16
9.3	Airinv Requirements	16
9.4	Basic Installation	16
9.5	Compilers and Options	17
9.6	Compiling For Multiple Architectures	17
9.7	Installation Names	18
9.8	Optional Features	19
9.9	Particular systems	19
9.10	Specifying the System Type	20
9.11	Sharing Defaults	20
9.12	Defining Variables	20
9.13	'cmake' Invocation	20
10	Linking with Airinv	24
10.1	Table of Contents	24
10.2	Introduction	25
10.3	Dependencies	25
10.3.1	StdAir	25
10.4	Using the pkg-config command	25

10.5 Using the airinv-config script	26
10.6 M4 macro for the GNU Autotools	26
10.7 Using Airinv with dynamic linking	26
11 Test Rules	26
11.1 The Test File	26
11.2 The Reference File	27
11.3 Testing IT++ Library	27
12 Users Guide	27
12.1 Table of Contents	27
12.2 Introduction	27
12.3 Get Started	28
12.3.1 Get the AirInv library	28
12.3.2 Build the AirInv project	28
12.3.3 Build and Run the Tests	28
12.3.4 Install the AirInv Project (Binaries, Documentation)	28
12.4 Input file of AirInv Project	29
12.5 The schedule BOM Tree	30
12.5.1 Build of the schedule BOM tree	30
12.5.2 Display of the schedule BOM tree	30
12.6 Exploring the Predefined BOM Tree	74
12.6.1 Airline Network BOM Tree	74
12.6.2 Airline Schedule BOM Tree	74
12.7 Extending the BOM Tree	74
12.8 The travel solution calculation procedure	74
13 Supported Systems	75
13.1 Table of Contents	75
13.2 Introduction	75
14 AirInv Supported Systems (Previous Releases)	75
14.1 AirInv 3.9.1	75
14.2 AirInv 3.9.0	75
14.3 AirInv 3.8.1	75
15 Tutorials	76
15.1 Table of Contents	76
15.2 Preparing the AirSched Project for Development	76
15.3 Your first networkBuilde	76
15.3.1 Summary of the different steps	76
15.3.2 Result of the Batch Program	76

15.4 Network building with an input file	77
15.4.1 How to build a network input file?	77
15.4.2 Building the BOM tree with an input file	78
15.4.3 Result of the Batch Program	78
16 Command-Line Test to Demonstrate How To Test the AirInv Project	78
17 Directory Hierarchy	82
17.1 Directories	82
18 Namespace Index	82
18.1 Namespace List	82
19 Class Index	83
19.1 Class Hierarchy	83
20 Class Index	89
20.1 Class List	89
21 File Index	94
21.1 File List	94
22 Directory Documentation	98
22.1 test/airinv/ Directory Reference	98
22.2 airinv/ Directory Reference	99
22.3 airinv/basic/ Directory Reference	99
22.4 airinv/batches/ Directory Reference	99
22.5 airinv/bom/ Directory Reference	99
22.6 airinv/ui/cmdline/ Directory Reference	100
22.7 airinv/command/ Directory Reference	100
22.8 airinv/config/ Directory Reference	101
22.9 airinv/factory/ Directory Reference	101
22.10airinv/server/ Directory Reference	101
22.11airinv/service/ Directory Reference	102
22.12test/ Directory Reference	102
22.13airinv/ui/ Directory Reference	102
22.14airinv/command/vault/ Directory Reference	102
23 Namespace Documentation	103
23.1 AIRINV Namespace Reference	103
23.1.1 Typedef Documentation	106
23.1.2 Variable Documentation	109
23.2 AIRINV::DCPParserHelper Namespace Reference	109

23.2.1	Variable Documentation	110
23.3	AIRINV::InventoryParserHelper Namespace Reference	111
23.3.1	Function Documentation	112
23.3.2	Variable Documentation	114
23.4	AIRINV::ScheduleParserHelper Namespace Reference	114
23.4.1	Function Documentation	115
23.4.2	Variable Documentation	116
23.5	stdair Namespace Reference	117
23.5.1	Detailed Description	117
23.6	swift Namespace Reference	117
23.6.1	Detailed Description	117
24	Class Documentation	118
24.1	AIRINV::AIRINV_Master_Service Class Reference	118
24.1.1	Detailed Description	118
24.1.2	Constructor & Destructor Documentation	118
24.1.3	Member Function Documentation	119
24.2	AIRINV::AIRINV_Master_ServiceContext Class Reference	123
24.2.1	Detailed Description	123
24.2.2	Friends And Related Function Documentation	123
24.3	AIRINV::AIRINV_Service Class Reference	123
24.3.1	Detailed Description	124
24.3.2	Constructor & Destructor Documentation	124
24.3.3	Member Function Documentation	125
24.4	AIRINV::AIRINV_ServiceContext Class Reference	129
24.4.1	Detailed Description	129
24.4.2	Friends And Related Function Documentation	129
24.5	AIRINV::AirInvServer Class Reference	129
24.5.1	Detailed Description	130
24.5.2	Constructor & Destructor Documentation	130
24.5.3	Member Function Documentation	130
24.6	AIRINV::BomAbstract Class Reference	130
24.6.1	Detailed Description	131
24.6.2	Constructor & Destructor Documentation	131
24.6.3	Member Function Documentation	131
24.6.4	Friends And Related Function Documentation	132
24.7	stdair::BomPropertyTree Struct Reference	132
24.7.1	Detailed Description	132
24.7.2	Member Function Documentation	132
24.7.3	Member Data Documentation	133

24.8 AIRINV::BomRootHelper Class Reference	133
24.8.1 Detailed Description	133
24.8.2 Member Function Documentation	134
24.9 AIRINV::BookingClassHelper Class Reference	134
24.9.1 Detailed Description	134
24.10 AIRINV::BookingClassStruct Struct Reference	134
24.10.1 Detailed Description	135
24.10.2 Constructor & Destructor Documentation	135
24.10.3 Member Function Documentation	135
24.10.4 Member Data Documentation	135
24.11 AIRINV::BookingException Class Reference	137
24.11.1 Detailed Description	138
24.12 AIRINV::BucketStruct Struct Reference	138
24.12.1 Detailed Description	138
24.12.2 Constructor & Destructor Documentation	138
24.12.3 Member Function Documentation	138
24.12.4 Member Data Documentation	139
24.13 CmdAbstract Class Reference	139
24.14 COMMAND Struct Reference	140
24.14.1 Detailed Description	140
24.14.2 Member Data Documentation	140
24.15 AIRINV::Connection Class Reference	141
24.15.1 Detailed Description	141
24.15.2 Constructor & Destructor Documentation	141
24.15.3 Member Function Documentation	141
24.16 AIRINV::DCPEventGenerator Class Reference	142
24.16.1 Detailed Description	142
24.16.2 Friends And Related Function Documentation	142
24.17 AIRINV::DCPEventStruct Struct Reference	143
24.17.1 Detailed Description	144
24.17.2 Constructor & Destructor Documentation	144
24.17.3 Member Function Documentation	144
24.17.4 Member Data Documentation	146
24.18 AIRINV::DCPParser Class Reference	149
24.18.1 Detailed Description	149
24.18.2 Member Function Documentation	149
24.19 AIRINV::DCPRuleFileParser Class Reference	150
24.19.1 Detailed Description	150
24.19.2 Constructor & Destructor Documentation	150
24.19.3 Member Function Documentation	150

24.20AIRINV::DCPParserHelper::DCPRuleParser Struct Reference	151
24.20.1 Detailed Description	152
24.20.2 Constructor & Destructor Documentation	153
24.20.3 Member Data Documentation	153
24.21AIRINV::DefaultMap Struct Reference	156
24.21.1 Detailed Description	156
24.21.2 Member Function Documentation	156
24.22AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT > Struct Template Reference	157
24.22.1 Detailed Description	158
24.22.2 Constructor & Destructor Documentation	158
24.22.3 Member Function Documentation	158
24.22.4 Member Data Documentation	159
24.23AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference	161
24.23.1 Detailed Description	163
24.23.2 Constructor & Destructor Documentation	163
24.23.3 Member Function Documentation	163
24.23.4 Member Data Documentation	163
24.24AIRINV::DCPParserHelper::doEndDCP Struct Reference	165
24.24.1 Detailed Description	166
24.24.2 Constructor & Destructor Documentation	166
24.24.3 Member Function Documentation	166
24.24.4 Member Data Documentation	166
24.25AIRINV::ScheduleParserHelper::doEndFlight Struct Reference	166
24.25.1 Detailed Description	167
24.25.2 Constructor & Destructor Documentation	167
24.25.3 Member Function Documentation	167
24.25.4 Member Data Documentation	167
24.26AIRINV::InventoryParserHelper::doEndFlightDate Struct Reference	168
24.26.1 Detailed Description	168
24.26.2 Constructor & Destructor Documentation	169
24.26.3 Member Function Documentation	169
24.26.4 Member Data Documentation	169
24.27enable_shared_from_this Class Reference	170
24.28AIRINV::FacAirinvMasterServiceContext Class Reference	170
24.28.1 Detailed Description	171
24.28.2 Constructor & Destructor Documentation	171
24.28.3 Member Function Documentation	171
24.29AIRINV::FacAirinvServiceContext Class Reference	172
24.29.1 Detailed Description	172

24.29.2 Constructor & Destructor Documentation	172
24.29.3 Member Function Documentation	173
24.30AIRINV::FacBomAbstract Class Reference	173
24.30.1 Detailed Description	174
24.30.2 Member Typedef Documentation	174
24.30.3 Constructor & Destructor Documentation	174
24.30.4 Member Function Documentation	174
24.30.5 Friends And Related Function Documentation	175
24.30.6 Member Data Documentation	175
24.31 FacServiceAbstract Class Reference	175
24.32AIRINV::FacServiceAbstract Class Reference	175
24.32.1 Detailed Description	176
24.32.2 Member Typedef Documentation	176
24.32.3 Constructor & Destructor Documentation	176
24.32.4 Member Function Documentation	176
24.32.5 Member Data Documentation	177
24.33AIRINV::FacSupervisor Class Reference	177
24.33.1 Detailed Description	177
24.33.2 Member Typedef Documentation	177
24.33.3 Constructor & Destructor Documentation	178
24.33.4 Member Function Documentation	178
24.34AIRINV::FareFamilyStruct Struct Reference	179
24.34.1 Detailed Description	180
24.34.2 Constructor & Destructor Documentation	180
24.34.3 Member Function Documentation	180
24.34.4 Member Data Documentation	180
24.35FileNotFoundException Class Reference	181
24.36AIRINV::FlightDateDuplicationException Class Reference	181
24.36.1 Detailed Description	181
24.36.2 Constructor & Destructor Documentation	181
24.37AIRINV::FlightDateHelper Class Reference	182
24.37.1 Detailed Description	182
24.37.2 Member Function Documentation	182
24.38AIRINV::FlightDateStruct Struct Reference	183
24.38.1 Detailed Description	184
24.38.2 Constructor & Destructor Documentation	184
24.38.3 Member Function Documentation	184
24.38.4 Member Data Documentation	185
24.39AIRINV::FlightPeriodFileParser Class Reference	188
24.39.1 Detailed Description	189

24.39.2 Constructor & Destructor Documentation	189
24.39.3 Member Function Documentation	189
24.40AIRINV::ScheduleParserHelper::FlightPeriodParser Struct Reference	189
24.40.1 Detailed Description	190
24.40.2 Constructor & Destructor Documentation	190
24.40.3 Member Data Documentation	190
24.41AIRINV::FlightPeriodStruct Struct Reference	191
24.41.1 Detailed Description	192
24.41.2 Constructor & Destructor Documentation	192
24.41.3 Member Function Documentation	192
24.41.4 Member Data Documentation	194
24.42AIRINV::FlightRequestStatus Struct Reference	196
24.42.1 Detailed Description	197
24.42.2 Member Enumeration Documentation	197
24.42.3 Constructor & Destructor Documentation	197
24.42.4 Member Function Documentation	197
24.43AIRINV::FlightTypeCode Struct Reference	198
24.43.1 Detailed Description	199
24.43.2 Member Enumeration Documentation	199
24.43.3 Constructor & Destructor Documentation	199
24.43.4 Member Function Documentation	199
24.44AIRINV::FlightVisibilityCode Struct Reference	200
24.44.1 Detailed Description	200
24.44.2 Member Enumeration Documentation	201
24.44.3 Constructor & Destructor Documentation	201
24.44.4 Member Function Documentation	201
24.45grammar Class Reference	202
24.46grammar Class Reference	202
24.47AIRINV::GuillotineBlockHelper Class Reference	202
24.47.1 Detailed Description	202
24.47.2 Member Function Documentation	203
24.48AIRINV::header Struct Reference	203
24.48.1 Detailed Description	203
24.48.2 Member Data Documentation	203
24.49AIRINV::InventoryBuilder Class Reference	203
24.49.1 Detailed Description	204
24.49.2 Friends And Related Function Documentation	204
24.50AIRINV::InventoryFileParser Class Reference	204
24.50.1 Detailed Description	204
24.50.2 Constructor & Destructor Documentation	204

24.50.3 Member Function Documentation	205
24.51 AIRINV::InventoryFileParsingFailedException Class Reference	205
24.51.1 Detailed Description	205
24.51.2 Constructor & Destructor Documentation	205
24.52 AIRINV::InventoryGenerator Class Reference	205
24.52.1 Detailed Description	206
24.52.2 Friends And Related Function Documentation	206
24.53 AIRINV::InventoryHelper Class Reference	206
24.53.1 Detailed Description	207
24.53.2 Member Function Documentation	207
24.54 AIRINV::InventoryInputFileNotFoundException Class Reference	208
24.54.1 Detailed Description	208
24.54.2 Constructor & Destructor Documentation	208
24.55 AIRINV::InventoryManager Class Reference	208
24.55.1 Detailed Description	209
24.55.2 Member Function Documentation	209
24.55.3 Friends And Related Function Documentation	210
24.56 AIRINV::InventoryParser Class Reference	210
24.56.1 Detailed Description	210
24.56.2 Member Function Documentation	210
24.57 AIRINV::InventoryParserHelper::InventoryParser Struct Reference	211
24.57.1 Detailed Description	211
24.57.2 Constructor & Destructor Documentation	211
24.57.3 Member Data Documentation	212
24.58 InventoryTestSuite Class Reference	212
24.58.1 Detailed Description	212
24.58.2 Constructor & Destructor Documentation	212
24.58.3 Member Function Documentation	213
24.58.4 Member Data Documentation	213
24.59 AIRINV::LegCabinHelper Class Reference	213
24.59.1 Detailed Description	213
24.60 AIRINV::LegCabinStruct Struct Reference	213
24.60.1 Detailed Description	214
24.60.2 Member Function Documentation	214
24.60.3 Member Data Documentation	214
24.61 AIRINV::LegStruct Struct Reference	216
24.61.1 Detailed Description	216
24.61.2 Constructor & Destructor Documentation	217
24.61.3 Member Function Documentation	217
24.61.4 Member Data Documentation	217

24.62noncopyable Class Reference	218
24.63ObjectCreationgDuplicationException Class Reference	219
24.64ParserException Class Reference	219
24.65AIRINV::InventoryParserHelper::ParserSemanticAction Struct Reference	219
24.65.1 Detailed Description	220
24.65.2 Constructor & Destructor Documentation	220
24.65.3 Member Data Documentation	220
24.66AIRINV::ScheduleParserHelper::ParserSemanticAction Struct Reference	221
24.66.1 Detailed Description	222
24.66.2 Constructor & Destructor Documentation	222
24.66.3 Member Data Documentation	222
24.67AIRINV::DCPPParserHelper::ParserSemanticAction Struct Reference	223
24.67.1 Detailed Description	224
24.67.2 Constructor & Destructor Documentation	224
24.67.3 Member Data Documentation	224
24.68ParsingFileFailedException Class Reference	225
24.69AIRINV::Reply Struct Reference	225
24.69.1 Detailed Description	225
24.69.2 Member Function Documentation	226
24.69.3 Member Data Documentation	226
24.70AIRINV::Request Struct Reference	226
24.70.1 Detailed Description	226
24.70.2 Member Function Documentation	227
24.70.3 Member Data Documentation	227
24.71AIRINV::RequestHandler Class Reference	227
24.71.1 Detailed Description	228
24.71.2 Constructor & Destructor Documentation	228
24.71.3 Member Function Documentation	228
24.72AIRINV::RequestParser Class Reference	228
24.72.1 Detailed Description	229
24.72.2 Constructor & Destructor Documentation	229
24.72.3 Member Function Documentation	229
24.73RootException Class Reference	229
24.74AIRINV::ScheduleFileParsingFailedException Class Reference	230
24.74.1 Detailed Description	230
24.74.2 Constructor & Destructor Documentation	230
24.75AIRINV::ScheduleInputFileNotFoundException Class Reference	230
24.75.1 Detailed Description	230
24.75.2 Constructor & Destructor Documentation	231
24.76AIRINV::ScheduleParser Class Reference	231

24.76.1 Detailed Description	231
24.76.2 Member Function Documentation	231
24.77AIRINV::SegmentCabinHelper Class Reference	232
24.77.1 Detailed Description	232
24.77.2 Member Function Documentation	232
24.78AIRINV::SegmentCabinStruct Struct Reference	233
24.78.1 Detailed Description	234
24.78.2 Member Function Documentation	234
24.78.3 Member Data Documentation	234
24.79AIRINV::SegmentDateHelper Class Reference	235
24.79.1 Detailed Description	235
24.79.2 Member Function Documentation	235
24.80AIRINV::SegmentDateNotFoundException Class Reference	236
24.80.1 Detailed Description	236
24.80.2 Constructor & Destructor Documentation	236
24.81AIRINV::SegmentStruct Struct Reference	236
24.81.1 Detailed Description	237
24.81.2 Member Function Documentation	237
24.81.3 Member Data Documentation	237
24.82ServiceAbstract Class Reference	238
24.83AIRINV::ServiceAbstract Class Reference	238
24.83.1 Detailed Description	239
24.83.2 Constructor & Destructor Documentation	239
24.83.3 Member Function Documentation	239
24.84swift::SKeymap Class Reference	240
24.84.1 Detailed Description	240
24.84.2 Constructor & Destructor Documentation	240
24.84.3 Member Function Documentation	241
24.84.4 Friends And Related Function Documentation	241
24.85swift::SReadline Class Reference	242
24.85.1 Detailed Description	242
24.85.2 Constructor & Destructor Documentation	243
24.85.3 Member Function Documentation	243
24.86AIRINV::InventoryParserHelper::storeACP Struct Reference	246
24.86.1 Detailed Description	247
24.86.2 Constructor & Destructor Documentation	247
24.86.3 Member Function Documentation	247
24.86.4 Member Data Documentation	247
24.87AIRINV::DCPParserHelper::storeAdvancePurchase Struct Reference	248
24.87.1 Detailed Description	248

24.87.2 Constructor & Destructor Documentation	248
24.87.3 Member Function Documentation	248
24.87.4 Member Data Documentation	249
24.88AIRINV::InventoryParserHelper::storeAirlineCode Struct Reference	249
24.88.1 Detailed Description	249
24.88.2 Constructor & Destructor Documentation	250
24.88.3 Member Function Documentation	250
24.88.4 Member Data Documentation	250
24.89AIRINV::ScheduleParserHelper::storeAirlineCode Struct Reference	251
24.89.1 Detailed Description	251
24.89.2 Constructor & Destructor Documentation	251
24.89.3 Member Function Documentation	251
24.89.4 Member Data Documentation	252
24.90AIRINV::DCPPParserHelper::storeAirlineCode Struct Reference	252
24.90.1 Detailed Description	252
24.90.2 Constructor & Destructor Documentation	253
24.90.3 Member Function Documentation	253
24.90.4 Member Data Documentation	253
24.91AIRINV::InventoryParserHelper::storeAU Struct Reference	253
24.91.1 Detailed Description	254
24.91.2 Constructor & Destructor Documentation	254
24.91.3 Member Function Documentation	254
24.91.4 Member Data Documentation	254
24.92AIRINV::InventoryParserHelper::storeBoardingDate Struct Reference	255
24.92.1 Detailed Description	255
24.92.2 Constructor & Destructor Documentation	255
24.92.3 Member Function Documentation	255
24.92.4 Member Data Documentation	256
24.93AIRINV::InventoryParserHelper::storeBoardingTime Struct Reference	256
24.93.1 Detailed Description	257
24.93.2 Constructor & Destructor Documentation	257
24.93.3 Member Function Documentation	257
24.93.4 Member Data Documentation	257
24.94AIRINV::ScheduleParserHelper::storeBoardingTime Struct Reference	258
24.94.1 Detailed Description	258
24.94.2 Constructor & Destructor Documentation	259
24.94.3 Member Function Documentation	259
24.94.4 Member Data Documentation	259
24.95AIRINV::InventoryParserHelper::storeBookingCounter Struct Reference	259
24.95.1 Detailed Description	260

24.95.2 Constructor & Destructor Documentation	260
24.95.3 Member Function Documentation	260
24.95.4 Member Data Documentation	260
24.96AIRINV::InventoryParserHelper::storeBucketAvaibility Struct Reference	261
24.96.1 Detailed Description	261
24.96.2 Constructor & Destructor Documentation	261
24.96.3 Member Function Documentation	262
24.96.4 Member Data Documentation	262
24.97AIRINV::DCPParserHelper::storeCabinCode Struct Reference	262
24.97.1 Detailed Description	263
24.97.2 Constructor & Destructor Documentation	263
24.97.3 Member Function Documentation	263
24.97.4 Member Data Documentation	263
24.98AIRINV::ScheduleParserHelper::storeCapacity Struct Reference	264
24.98.1 Detailed Description	264
24.98.2 Constructor & Destructor Documentation	264
24.98.3 Member Function Documentation	264
24.98.4 Member Data Documentation	265
24.99AIRINV::DCPParserHelper::storeChangeFees Struct Reference	265
24.99.1 Detailed Description	265
24.99.2 Constructor & Destructor Documentation	266
24.99.3 Member Function Documentation	266
24.99.4 Member Data Documentation	266
24.100AIRINV::DCPParserHelper::storeChannel Struct Reference	266
24.100.1 Detailed Description	267
24.100.2 Constructor & Destructor Documentation	267
24.100.3 Member Function Documentation	267
24.100.4 Member Data Documentation	267
24.101AIRINV::DCPParserHelper::storeClass Struct Reference	267
24.101.1 Detailed Description	268
24.101.2 Constructor & Destructor Documentation	268
24.101.3 Member Function Documentation	268
24.101.4 Member Data Documentation	268
24.102AIRINV::InventoryParserHelper::storeClassAvailability Struct Reference	269
24.102.1 Detailed Description	269
24.102.2 Constructor & Destructor Documentation	269
24.102.3 Member Function Documentation	269
24.102.4 Member Data Documentation	270
24.103AIRINV::InventoryParserHelper::storeClassCode Struct Reference	270
24.103.1 Detailed Description	271

24.103.2	Constructor & Destructor Documentation	271
24.103.3	Member Function Documentation	271
24.103.4	Member Data Documentation	271
24.104	AIRINV::ScheduleParserHelper::storeClasses Struct Reference	272
24.104.1	Detailed Description	272
24.104.2	Constructor & Destructor Documentation	272
24.104.3	Member Function Documentation	273
24.104.4	Member Data Documentation	273
24.105	AIRINV::InventoryParserHelper::storeClassETB Struct Reference	273
24.105.1	Detailed Description	274
24.105.2	Constructor & Destructor Documentation	274
24.105.3	Member Function Documentation	274
24.105.4	Member Data Documentation	274
24.106	AIRINV::InventoryParserHelper::storeCumulatedProtection Struct Reference	275
24.106.1	Detailed Description	275
24.106.2	Constructor & Destructor Documentation	275
24.106.3	Member Function Documentation	275
24.106.4	Member Data Documentation	276
24.107	AIRINV::DCPPParserHelper::storeDateRangeEnd Struct Reference	276
24.107.1	Detailed Description	277
24.107.2	Constructor & Destructor Documentation	277
24.107.3	Member Function Documentation	277
24.107.4	Member Data Documentation	277
24.108	AIRINV::ScheduleParserHelper::storeDateRangeEnd Struct Reference	278
24.108.1	Detailed Description	278
24.108.2	Constructor & Destructor Documentation	278
24.108.3	Member Function Documentation	278
24.108.4	Member Data Documentation	279
24.109	AIRINV::ScheduleParserHelper::storeDateRangeStart Struct Reference	279
24.109.1	Detailed Description	279
24.109.2	Constructor & Destructor Documentation	280
24.109.3	Member Function Documentation	280
24.109.4	Member Data Documentation	280
24.110	AIRINV::DCPPParserHelper::storeDateRangeStart Struct Reference	280
24.110.1	Detailed Description	281
24.110.2	Constructor & Destructor Documentation	281
24.110.3	Member Function Documentation	281
24.110.4	Member Data Documentation	281
24.111	AIRINV::DCPPParserHelper::storeDCP Struct Reference	282
24.111.1	Detailed Description	282

24.111.2	Constructor & Destructor Documentation	282
24.111.3	Member Function Documentation	282
24.111.4	Member Data Documentation	282
24.112	AIRINV::DCPParserHelper::storeDCPID Struct Reference	283
24.112.1	Detailed Description	283
24.112.2	Constructor & Destructor Documentation	283
24.112.3	Member Function Documentation	283
24.112.4	Member Data Documentation	284
24.113	AIRINV::DCPParserHelper::storeDestination Struct Reference	284
24.113.1	Detailed Description	284
24.113.2	Constructor & Destructor Documentation	285
24.113.3	Member Function Documentation	285
24.113.4	Member Data Documentation	285
24.114	AIRINV::ScheduleParserHelper::storeDow Struct Reference	285
24.114.1	Detailed Description	286
24.114.2	Constructor & Destructor Documentation	286
24.114.3	Member Function Documentation	286
24.114.4	Member Data Documentation	286
24.115	AIRINV::ScheduleParserHelper::storeElapsedTime Struct Reference	286
24.115.1	Detailed Description	287
24.115.2	Constructor & Destructor Documentation	287
24.115.3	Member Function Documentation	287
24.115.4	Member Data Documentation	287
24.116	AIRINV::DCPParserHelper::storeEndRangeTime Struct Reference	288
24.116.1	Detailed Description	288
24.116.2	Constructor & Destructor Documentation	288
24.116.3	Member Function Documentation	288
24.116.4	Member Data Documentation	289
24.117	AIRINV::InventoryParserHelper::storeETB Struct Reference	289
24.117.1	Detailed Description	289
24.117.2	Constructor & Destructor Documentation	290
24.117.3	Member Function Documentation	290
24.117.4	Member Data Documentation	290
24.118	AIRINV::InventoryParserHelper::storeFamilyCode Struct Reference	291
24.118.1	Detailed Description	291
24.118.2	Constructor & Destructor Documentation	291
24.118.3	Member Function Documentation	291
24.118.4	Member Data Documentation	292
24.119	AIRINV::ScheduleParserHelper::storeFamilyCode Struct Reference	292
24.119.1	Detailed Description	293

24.119.2	Constructor & Destructor Documentation	293
24.119.3	Member Function Documentation	293
24.119.4	Member Data Documentation	293
24.120	AIRINV::InventoryParserHelper::storeFClasses Struct Reference	294
24.120.1	Detailed Description	294
24.120.2	Constructor & Destructor Documentation	294
24.120.3	Member Function Documentation	294
24.120.4	Member Data Documentation	294
24.121	AIRINV::ScheduleParserHelper::storeFClasses Struct Reference	295
24.121.1	Detailed Description	296
24.121.2	Constructor & Destructor Documentation	296
24.121.3	Member Function Documentation	296
24.121.4	Member Data Documentation	296
24.122	AIRINV::InventoryParserHelper::storeFlightDate Struct Reference	296
24.122.1	Detailed Description	297
24.122.2	Constructor & Destructor Documentation	297
24.122.3	Member Function Documentation	297
24.122.4	Member Data Documentation	297
24.123	AIRINV::InventoryParserHelper::storeFlightNumber Struct Reference	298
24.123.1	Detailed Description	298
24.123.2	Constructor & Destructor Documentation	299
24.123.3	Member Function Documentation	299
24.123.4	Member Data Documentation	299
24.124	AIRINV::ScheduleParserHelper::storeFlightNumber Struct Reference	300
24.124.1	Detailed Description	300
24.124.2	Constructor & Destructor Documentation	300
24.124.3	Member Function Documentation	300
24.124.4	Member Data Documentation	300
24.125	AIRINV::InventoryParserHelper::storeFlightTypeCode Struct Reference	301
24.125.1	Detailed Description	301
24.125.2	Constructor & Destructor Documentation	301
24.125.3	Member Function Documentation	302
24.125.4	Member Data Documentation	302
24.126	AIRINV::InventoryParserHelper::storeFlightVisibilityCode Struct Reference	302
24.126.1	Detailed Description	303
24.126.2	Constructor & Destructor Documentation	303
24.126.3	Member Function Documentation	303
24.126.4	Member Data Documentation	303
24.127	AIRINV::InventoryParserHelper::storeGAV Struct Reference	304
24.127.1	Detailed Description	304

24.127.2	Constructor & Destructor Documentation	305
24.127.3	Member Function Documentation	305
24.127.4	Member Data Documentation	305
24.128	AIRINV::InventoryParserHelper::storeLegBoardingPoint Struct Reference	306
24.128.1	Detailed Description	306
24.128.2	Constructor & Destructor Documentation	306
24.128.3	Member Function Documentation	306
24.128.4	Member Data Documentation	307
24.129	AIRINV::ScheduleParserHelper::storeLegBoardingPoint Struct Reference	307
24.129.1	Detailed Description	308
24.129.2	Constructor & Destructor Documentation	308
24.129.3	Member Function Documentation	308
24.129.4	Member Data Documentation	308
24.130	AIRINV::ScheduleParserHelper::storeLegCabinCode Struct Reference	309
24.130.1	Detailed Description	309
24.130.2	Constructor & Destructor Documentation	309
24.130.3	Member Function Documentation	309
24.130.4	Member Data Documentation	309
24.131	AIRINV::InventoryParserHelper::storeLegCabinCode Struct Reference	310
24.131.1	Detailed Description	310
24.131.2	Constructor & Destructor Documentation	310
24.131.3	Member Function Documentation	311
24.131.4	Member Data Documentation	311
24.132	AIRINV::ScheduleParserHelper::storeLegOffPoint Struct Reference	311
24.132.1	Detailed Description	312
24.132.2	Constructor & Destructor Documentation	312
24.132.3	Member Function Documentation	312
24.132.4	Member Data Documentation	312
24.133	AIRINV::InventoryParserHelper::storeLegOffPoint Struct Reference	313
24.133.1	Detailed Description	313
24.133.2	Constructor & Destructor Documentation	313
24.133.3	Member Function Documentation	313
24.133.4	Member Data Documentation	314
24.134	AIRINV::DCPPParserHelper::storeMinimumStay Struct Reference	314
24.134.1	Detailed Description	315
24.134.2	Constructor & Destructor Documentation	315
24.134.3	Member Function Documentation	315
24.134.4	Member Data Documentation	315
24.135	AIRINV::InventoryParserHelper::storeNAV Struct Reference	316
24.135.1	Detailed Description	316

24.135.2	Constructor & Destructor Documentation	316
24.135.3	Member Function Documentation	316
24.135.4	Member Data Documentation	317
24.136	AIRINV::InventoryParserHelper::storeNbOfBkgs Struct Reference	317
24.136.1	Detailed Description	318
24.136.2	Constructor & Destructor Documentation	318
24.136.3	Member Function Documentation	318
24.136.4	Member Data Documentation	318
24.137	AIRINV::InventoryParserHelper::storeNbOfGroupBkgs Struct Reference	319
24.137.1	Detailed Description	319
24.137.2	Constructor & Destructor Documentation	319
24.137.3	Member Function Documentation	320
24.137.4	Member Data Documentation	320
24.138	AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs Struct Reference	320
24.138.1	Detailed Description	321
24.138.2	Constructor & Destructor Documentation	321
24.138.3	Member Function Documentation	321
24.138.4	Member Data Documentation	321
24.139	AIRINV::InventoryParserHelper::storeNbOfStaffBkgs Struct Reference	322
24.139.1	Detailed Description	322
24.139.2	Constructor & Destructor Documentation	323
24.139.3	Member Function Documentation	323
24.139.4	Member Data Documentation	323
24.140	AIRINV::InventoryParserHelper::storeNbOfWLBkgs Struct Reference	324
24.140.1	Detailed Description	324
24.140.2	Constructor & Destructor Documentation	324
24.140.3	Member Function Documentation	324
24.140.4	Member Data Documentation	324
24.141	AIRINV::InventoryParserHelper::storeNego Struct Reference	325
24.141.1	Detailed Description	326
24.141.2	Constructor & Destructor Documentation	326
24.141.3	Member Function Documentation	326
24.141.4	Member Data Documentation	326
24.142	AIRINV::DCPPParserHelper::storeNonRefundable Struct Reference	327
24.142.1	Detailed Description	327
24.142.2	Constructor & Destructor Documentation	327
24.142.3	Member Function Documentation	327
24.142.4	Member Data Documentation	328
24.143	AIRINV::InventoryParserHelper::storeNoShow Struct Reference	328
24.143.1	Detailed Description	328

24.143.2	Constructor & Destructor Documentation	328
24.143.3	Member Function Documentation	329
24.143.4	Member Data Documentation	329
24.144	AIRINV::InventoryParserHelper::storeOffDate Struct Reference	329
24.144.1	Detailed Description	330
24.144.2	Constructor & Destructor Documentation	330
24.144.3	Member Function Documentation	330
24.144.4	Member Data Documentation	330
24.145	AIRINV::InventoryParserHelper::storeOffTime Struct Reference	331
24.145.1	Detailed Description	331
24.145.2	Constructor & Destructor Documentation	332
24.145.3	Member Function Documentation	332
24.145.4	Member Data Documentation	332
24.146	AIRINV::ScheduleParserHelper::storeOffTime Struct Reference	333
24.146.1	Detailed Description	333
24.146.2	Constructor & Destructor Documentation	333
24.146.3	Member Function Documentation	333
24.146.4	Member Data Documentation	333
24.147	AIRINV::DCPParserHelper::storeOrigin Struct Reference	334
24.147.1	Detailed Description	334
24.147.2	Constructor & Destructor Documentation	334
24.147.3	Member Function Documentation	335
24.147.4	Member Data Documentation	335
24.148	AIRINV::InventoryParserHelper::storeOverbooking Struct Reference	335
24.148.1	Detailed Description	336
24.148.2	Constructor & Destructor Documentation	336
24.148.3	Member Function Documentation	336
24.148.4	Member Data Documentation	336
24.149	AIRINV::InventoryParserHelper::storeParentClassCode Struct Reference	337
24.149.1	Detailed Description	337
24.149.2	Constructor & Destructor Documentation	337
24.149.3	Member Function Documentation	337
24.149.4	Member Data Documentation	338
24.150	AIRINV::InventoryParserHelper::storeParentSubclassCode Struct Reference	338
24.150.1	Detailed Description	339
24.150.2	Constructor & Destructor Documentation	339
24.150.3	Member Function Documentation	339
24.150.4	Member Data Documentation	339
24.151	AIRINV::DCPParserHelper::storePOS Struct Reference	340
24.151.1	Detailed Description	340

24.151.2	Constructor & Destructor Documentation	340
24.151.3	Member Function Documentation	341
24.151.4	Member Data Documentation	341
24.152	AIRINV::InventoryParserHelper::storeProtection Struct Reference	341
24.152.1	Detailed Description	342
24.152.2	Constructor & Destructor Documentation	342
24.152.3	Member Function Documentation	342
24.152.4	Member Data Documentation	342
24.153	AIRINV::InventoryParserHelper::storeRevenueAvailability Struct Reference	343
24.153.1	Detailed Description	343
24.153.2	Constructor & Destructor Documentation	343
24.153.3	Member Function Documentation	343
24.153.4	Member Data Documentation	344
24.154	AIRINV::InventoryParserHelper::storeSaleableCapacity Struct Reference	344
24.154.1	Detailed Description	345
24.154.2	Constructor & Destructor Documentation	345
24.154.3	Member Function Documentation	345
24.154.4	Member Data Documentation	345
24.155	AIRINV::DCPParserHelper::storeSaturdayStay Struct Reference	346
24.155.1	Detailed Description	346
24.155.2	Constructor & Destructor Documentation	346
24.155.3	Member Function Documentation	347
24.155.4	Member Data Documentation	347
24.156	AIRINV::InventoryParserHelper::storeSeatIndex Struct Reference	347
24.156.1	Detailed Description	348
24.156.2	Constructor & Destructor Documentation	348
24.156.3	Member Function Documentation	348
24.156.4	Member Data Documentation	348
24.157	AIRINV::InventoryParserHelper::storeSegmentAvailability Struct Reference	349
24.157.1	Detailed Description	349
24.157.2	Constructor & Destructor Documentation	349
24.157.3	Member Function Documentation	349
24.157.4	Member Data Documentation	350
24.158	AIRINV::InventoryParserHelper::storeSegmentBoardingPoint Struct Reference	350
24.158.1	Detailed Description	351
24.158.2	Constructor & Destructor Documentation	351
24.158.3	Member Function Documentation	351
24.158.4	Member Data Documentation	351
24.159	AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference	352
24.159.1	Detailed Description	352

24.159.2	Constructor & Destructor Documentation	352
24.159.3	Member Function Documentation	353
24.159.4	Member Data Documentation	353
24.160	AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter Struct Reference	353
24.160.1	Detailed Description	354
24.160.2	Constructor & Destructor Documentation	354
24.160.3	Member Function Documentation	354
24.160.4	Member Data Documentation	354
24.161	AIRINV::InventoryParserHelper::storeSegmentCabinCode Struct Reference	355
24.161.1	Detailed Description	355
24.161.2	Constructor & Destructor Documentation	355
24.161.3	Member Function Documentation	356
24.161.4	Member Data Documentation	356
24.162	AIRINV::ScheduleParserHelper::storeSegmentCabinCode Struct Reference	357
24.162.1	Detailed Description	357
24.162.2	Constructor & Destructor Documentation	357
24.162.3	Member Function Documentation	357
24.162.4	Member Data Documentation	357
24.163	AIRINV::InventoryParserHelper::storeSegmentOffPoint Struct Reference	358
24.163.1	Detailed Description	358
24.163.2	Constructor & Destructor Documentation	358
24.163.3	Member Function Documentation	359
24.163.4	Member Data Documentation	359
24.164	AIRINV::ScheduleParserHelper::storeSegmentOffPoint Struct Reference	359
24.164.1	Detailed Description	360
24.164.2	Constructor & Destructor Documentation	360
24.164.3	Member Function Documentation	360
24.164.4	Member Data Documentation	360
24.165	AIRINV::ScheduleParserHelper::storeSegmentSpecificity Struct Reference	361
24.165.1	Detailed Description	361
24.165.2	Constructor & Destructor Documentation	361
24.165.3	Member Function Documentation	361
24.165.4	Member Data Documentation	362
24.166	AIRINV::InventoryParserHelper::storeSnapshotDate Struct Reference	362
24.166.1	Detailed Description	362
24.166.2	Constructor & Destructor Documentation	363
24.166.3	Member Function Documentation	363
24.166.4	Member Data Documentation	363
24.167	AIRINV::DCPPParserHelper::storeStartRangeTime Struct Reference	364
24.167.1	Detailed Description	364

24.167.2	Constructor & Destructor Documentation	364
24.167.3	Member Function Documentation	364
24.167.4	Member Data Documentation	364
24.168	AIRINV::InventoryParserHelper::storeSubclassCode Struct Reference	365
24.168.1	Detailed Description	365
24.168.2	Constructor & Destructor Documentation	365
24.168.3	Member Function Documentation	366
24.168.4	Member Data Documentation	366
24.169	AIRINV::InventoryParserHelper::storeUPR Struct Reference	366
24.169.1	Detailed Description	367
24.169.2	Constructor & Destructor Documentation	367
24.169.3	Member Function Documentation	367
24.169.4	Member Data Documentation	367
24.170	AIRINV::InventoryParserHelper::storeYieldUpperRange Struct Reference	368
24.170.1	Detailed Description	368
24.170.2	Constructor & Destructor Documentation	369
24.170.3	Member Function Documentation	369
24.170.4	Member Data Documentation	369
24.171	StructAbstract Class Reference	370
24.172	TestFixture Class Reference	370
25	File Documentation	371
25.1	airinv/AIRINV_Master_Service.hpp File Reference	371
25.2	AIRINV_Master_Service.hpp	371
25.3	airinv/AIRINV_Service.hpp File Reference	373
25.4	AIRINV_Service.hpp	373
25.5	airinv/AIRINV_Types.hpp File Reference	375
25.6	AIRINV_Types.hpp	375
25.7	airinv/basic/BasConst.cpp File Reference	376
25.8	BasConst.cpp	376
25.9	airinv/basic/BasConst_AIRINV_Service.hpp File Reference	378
25.10	BasConst_AIRINV_Service.hpp	378
25.11	airinv/basic/BasConst_Curves.hpp File Reference	378
25.12	BasConst_Curves.hpp	378
25.13	airinv/basic/BasConst_General.hpp File Reference	379
25.14	BasConst_General.hpp	379
25.15	airinv/basic/BasParserTypes.hpp File Reference	379
25.16	BasParserTypes.hpp	380
25.17	airinv/basic/FlightRequestStatus.cpp File Reference	381
25.18	FlightRequestStatus.cpp	381

25.19airinv/basic/FlightTypeCode.cpp File Reference	383
25.20FlightTypeCode.cpp	383
25.21airinv/basic/FlightTypeCode.hpp File Reference	384
25.22FlightTypeCode.hpp	384
25.23airinv/basic/FlightVisibilityCode.cpp File Reference	385
25.24FlightVisibilityCode.cpp	385
25.25airinv/basic/FlightVisibilityCode.hpp File Reference	386
25.26FlightVisibilityCode.hpp	387
25.27airinv/batches/airinv_parseInventory.cpp File Reference	387
25.28airinv_parseInventory.cpp	387
25.29airinv/batches/parseInventory.cpp File Reference	391
25.30parseInventory.cpp	391
25.31airinv/bom/AirportList.hpp File Reference	395
25.32AirportList.hpp	395
25.33airinv/bom/BomAbstract.cpp File Reference	396
25.34BomAbstract.cpp	396
25.35airinv/bom/BomAbstract.hpp File Reference	396
25.35.1 Function Documentation	396
25.36BomAbstract.hpp	397
25.37airinv/bom/BomRootHelper.cpp File Reference	398
25.38BomRootHelper.cpp	398
25.39airinv/bom/BomRootHelper.hpp File Reference	398
25.40BomRootHelper.hpp	398
25.41airinv/bom/BookingClassHelper.cpp File Reference	399
25.42BookingClassHelper.cpp	399
25.43airinv/bom/BookingClassHelper.hpp File Reference	399
25.44BookingClassHelper.hpp	399
25.45airinv/bom/BookingClassStruct.cpp File Reference	400
25.46BookingClassStruct.cpp	400
25.47airinv/bom/BookingClassStruct.hpp File Reference	401
25.48BookingClassStruct.hpp	401
25.49airinv/bom/BucketStruct.cpp File Reference	402
25.50BucketStruct.cpp	402
25.51airinv/bom/BucketStruct.hpp File Reference	403
25.52BucketStruct.hpp	403
25.53airinv/bom/DCPEventStruct.cpp File Reference	404
25.54DCPEventStruct.cpp	404
25.55airinv/bom/DCPEventStruct.hpp File Reference	406
25.56DCPEventStruct.hpp	406
25.57airinv/bom/FareFamilyStruct.cpp File Reference	408

25.58FareFamilyStruct.cpp	408
25.59airinv/bom/FareFamilyStruct.hpp File Reference	409
25.60FareFamilyStruct.hpp	409
25.61 airinv/bom/FlightDateHelper.cpp File Reference	410
25.62FlightDateHelper.cpp	410
25.63airinv/bom/FlightDateHelper.hpp File Reference	411
25.64FlightDateHelper.hpp	411
25.65airinv/bom/FlightDateStruct.cpp File Reference	412
25.66FlightDateStruct.cpp	412
25.67 airinv/bom/FlightDateStruct.hpp File Reference	415
25.68FlightDateStruct.hpp	416
25.69airinv/bom/FlightPeriodStruct.cpp File Reference	417
25.70FlightPeriodStruct.cpp	417
25.71 airinv/bom/FlightPeriodStruct.hpp File Reference	420
25.72FlightPeriodStruct.hpp	421
25.73airinv/bom/GuillotineBlockHelper.cpp File Reference	422
25.74GuillotineBlockHelper.cpp	422
25.75airinv/bom/GuillotineBlockHelper.hpp File Reference	425
25.76GuillotineBlockHelper.hpp	425
25.77 airinv/bom/InventoryHelper.cpp File Reference	426
25.78InventoryHelper.cpp	426
25.79airinv/bom/InventoryHelper.hpp File Reference	431
25.80InventoryHelper.hpp	431
25.81 airinv/bom/LegCabinHelper.cpp File Reference	432
25.82LegCabinHelper.cpp	432
25.83airinv/bom/LegCabinHelper.hpp File Reference	432
25.84LegCabinHelper.hpp	432
25.85airinv/bom/LegCabinStruct.cpp File Reference	433
25.86LegCabinStruct.cpp	433
25.87 airinv/bom/LegCabinStruct.hpp File Reference	433
25.88LegCabinStruct.hpp	434
25.89airinv/bom/LegStruct.cpp File Reference	434
25.90LegStruct.cpp	435
25.91 airinv/bom/LegStruct.hpp File Reference	436
25.92LegStruct.hpp	436
25.93airinv/bom/SegmentCabinHelper.cpp File Reference	437
25.94SegmentCabinHelper.cpp	437
25.95airinv/bom/SegmentCabinHelper.hpp File Reference	439
25.96SegmentCabinHelper.hpp	440
25.97airinv/bom/SegmentCabinStruct.cpp File Reference	440

25.98SegmentCabinStruct.cpp	441
25.99airinv/bom/SegmentCabinStruct.hpp File Reference	441
25.100SegmentCabinStruct.hpp	442
25.101airinv/bom/SegmentDateHelper.cpp File Reference	442
25.102SegmentDateHelper.cpp	442
25.103airinv/bom/SegmentDateHelper.hpp File Reference	444
25.104SegmentDateHelper.hpp	444
25.105airinv/bom/SegmentStruct.cpp File Reference	444
25.106SegmentStruct.cpp	445
25.107airinv/bom/SegmentStruct.hpp File Reference	445
25.108SegmentStruct.hpp	446
25.109airinv/command/InventoryBuilder.cpp File Reference	446
25.110InventoryBuilder.cpp	447
25.111airinv/command/InventoryBuilder.hpp File Reference	451
25.112InventoryBuilder.hpp	451
25.113airinv/command/InventoryGenerator.cpp File Reference	452
25.114InventoryGenerator.cpp	453
25.115airinv/command/InventoryGenerator.hpp File Reference	457
25.116InventoryGenerator.hpp	457
25.117airinv/command/InventoryManager.cpp File Reference	458
25.118InventoryManager.cpp	459
25.119airinv/command/InventoryManager.hpp File Reference	472
25.120InventoryManager.hpp	473
25.121airinv/command/InventoryParser.cpp File Reference	474
25.122InventoryParser.cpp	474
25.123airinv/command/InventoryParser.hpp File Reference	475
25.124InventoryParser.hpp	475
25.125airinv/command/InventoryParserHelper.cpp File Reference	475
25.126InventoryParserHelper.cpp	476
25.127airinv/command/InventoryParserHelper.hpp File Reference	489
25.128InventoryParserHelper.hpp	491
25.129airinv/command/ScheduleParser.cpp File Reference	495
25.130ScheduleParser.cpp	495
25.131airinv/command/ScheduleParser.hpp File Reference	496
25.132ScheduleParser.hpp	496
25.133airinv/command/ScheduleParserHelper.cpp File Reference	497
25.134ScheduleParserHelper.cpp	497
25.135airinv/command/ScheduleParserHelper.hpp File Reference	505
25.136ScheduleParserHelper.hpp	506
25.137airinv/command/vault/DCPEventGenerator.cpp File Reference	508

25.138	DCPEventGenerator.cpp	509
25.139	airinv/command/vault/DCPEventGenerator.hpp File Reference	510
25.140	DCPEventGenerator.hpp	510
25.141	airinv/command/vault/DCPParser.cpp File Reference	510
25.142	DCPParser.cpp	511
25.143	airinv/command/vault/DCPParser.hpp File Reference	511
25.144	DCPParser.hpp	511
25.145	airinv/command/vault/DCPParserHelper.cpp File Reference	512
25.146	DCPParserHelper.cpp	512
25.147	airinv/command/vault/DCPParserHelper.hpp File Reference	520
25.148	DCPParserHelper.hpp	521
25.149	airinv/config/airinv-paths.hpp File Reference	523
25.149	Define Documentation	524
25.150	airinv-paths.hpp	525
25.151	airinv/config/airinv-paths.hpp.in File Reference	525
25.151	Define Documentation	526
25.152	airinv-paths.hpp.in	527
25.153	airinv/factory/FacAirinvMasterServiceContext.cpp File Reference	527
25.154	FacAirinvMasterServiceContext.cpp	528
25.155	airinv/factory/FacAirinvMasterServiceContext.hpp File Reference	528
25.156	FacAirinvMasterServiceContext.hpp	528
25.157	airinv/factory/FacAirinvServiceContext.cpp File Reference	529
25.158	FacAirinvServiceContext.cpp	529
25.159	airinv/factory/FacAirinvServiceContext.hpp File Reference	530
25.160	FacAirinvServiceContext.hpp	530
25.161	airinv/factory/FacBomAbstract.cpp File Reference	531
25.162	FacBomAbstract.cpp	531
25.163	airinv/factory/FacBomAbstract.hpp File Reference	532
25.164	FacBomAbstract.hpp	532
25.165	airinv/factory/FacServiceAbstract.cpp File Reference	532
25.166	FacServiceAbstract.cpp	533
25.167	airinv/factory/FacServiceAbstract.hpp File Reference	533
25.168	FacServiceAbstract.hpp	533
25.169	airinv/factory/FacSupervisor.cpp File Reference	534
25.170	FacSupervisor.cpp	534
25.171	airinv/factory/FacSupervisor.hpp File Reference	535
25.172	FacSupervisor.hpp	535
25.173	airinv/FlightRequestStatus.hpp File Reference	536
25.174	FlightRequestStatus.hpp	536
25.175	airinv/server/AirInvClient.cpp File Reference	537

25.175. Function Documentation	537
25.176. AirInvClient.cpp	537
25.177. airinv/server/AirInvClient_ASIO.cpp File Reference	538
25.177. Function Documentation	538
25.178. AirInvClient_ASIO.cpp	538
25.179. airinv/server/AirInvServer.cpp File Reference	539
25.180. AirInvServer.cpp	539
25.181. airinv/server/AirInvServer.hpp File Reference	544
25.182. AirInvServer.hpp	544
25.183. airinv/server/AirInvServer_ASIO.cpp File Reference	545
25.184. AirInvServer_ASIO.cpp	545
25.185. airinv/server/BomPropertyTree.cpp File Reference	546
25.186. BomPropertyTree.cpp	547
25.187. airinv/server/BomPropertyTree.hpp File Reference	548
25.188. BomPropertyTree.hpp	548
25.189. airinv/server/Connection.cpp File Reference	549
25.190. Connection.cpp	549
25.191. airinv/server/Connection.hpp File Reference	550
25.192. Connection.hpp	551
25.193. airinv/server/header.cpp File Reference	551
25.194. header.hpp	552
25.195. airinv/server/posix_main.cpp File Reference	552
25.195. Function Documentation	552
25.196. posix_main.cpp	552
25.197. airinv/server/Reply.cpp File Reference	553
25.198. Reply.cpp	553
25.199. airinv/server/Reply.hpp File Reference	554
25.200. Reply.hpp	554
25.201. airinv/server/Request.cpp File Reference	554
25.202. Request.cpp	555
25.203. airinv/server/Request.hpp File Reference	555
25.204. Request.hpp	555
25.205. airinv/server/RequestHandler.cpp File Reference	556
25.206. RequestHandler.cpp	556
25.207. airinv/server/RequestHandler.hpp File Reference	557
25.208. RequestHandler.hpp	557
25.209. airinv/server/RequestParser.cpp File Reference	558
25.210. RequestParser.cpp	558
25.211. airinv/server/RequestParser.hpp File Reference	561
25.212. RequestParser.hpp	561

25.21	airinv/server/win_main.cpp File Reference	562
25.21	win_main.cpp	562
25.21	airinv/service/AIRINV_Master_Service.cpp File Reference	563
25.21	AIRINV_Master_Service.cpp	564
25.21	airinv/service/AIRINV_Master_ServiceContext.cpp File Reference	571
25.21	AIRINV_Master_ServiceContext.cpp	571
25.21	airinv/service/AIRINV_Master_ServiceContext.hpp File Reference	572
25.22	AIRINV_Master_ServiceContext.hpp	572
25.22	airinv/service/AIRINV_Service.cpp File Reference	573
25.22	AIRINV_Service.cpp	574
25.22	airinv/service/AIRINV_ServiceContext.cpp File Reference	582
25.22	AIRINV_ServiceContext.cpp	582
25.22	airinv/service/AIRINV_ServiceContext.hpp File Reference	583
25.22	AIRINV_ServiceContext.hpp	583
25.22	airinv/service/ServiceAbstract.cpp File Reference	584
25.22	ServiceAbstract.cpp	584
25.22	airinv/service/ServiceAbstract.hpp File Reference	585
25.229	Function Documentation	585
25.23	ServiceAbstract.hpp	585
25.23	airinv/ui/cmdline/airinv.cpp File Reference	586
25.23	airinv.cpp	586
25.23	airinv/ui/cmdline/readline_autocomp.hpp File Reference	598
25.233	Typedef Documentation	599
25.233	Function Documentation	599
25.233	Variable Documentation	601
25.23	readline_autocomp.hpp	602
25.23	airinv/ui/cmdline/SReadline.hpp File Reference	606
25.235	Detailed Description	606
25.23	SReadline.hpp	606
25.23	doc/local/authors.doc File Reference	612
25.23	doc/local/codingrules.doc File Reference	612
25.23	doc/local/copyright.doc File Reference	612
25.24	doc/local/documentation.doc File Reference	612
25.24	doc/local/features.doc File Reference	612
25.24	doc/local/help_wanted.doc File Reference	612
25.24	doc/local/howto_release.doc File Reference	612
25.24	doc/local/index.doc File Reference	612
25.24	doc/local/installation.doc File Reference	612
25.24	doc/local/linking.doc File Reference	612
25.24	doc/local/test.doc File Reference	612

doc/local/users_guide.doc File Reference	612
doc/local/verification.doc File Reference	612
doc/tutorial/tutorial.doc File Reference	612
test/airinv/InventoryTestSuite.cpp File Reference	612
InventoryTestSuite.cpp	612
test/airinv/InventoryTestSuite.hpp File Reference	616
25.253. Function Documentation	616
InventoryTestSuite.hpp	617

1 AirInv Documentation

1.1 Getting Started

- [Main features](#)
- [Installation](#)
- [Linking with AirInv](#)
- [Users Guide](#)
- [Tutorials](#)
- [Copyright and License](#)
- [Make a Difference](#)
- [Make a new release](#)
- [People](#)

1.2 AirInv at SourceForge

- [Project page](#)
- [Download AirInv](#)
- [Open a ticket for a bug or feature](#)
- [Mailing lists](#)
- [Forums](#)
 - [Discuss about Development issues](#)
 - [Ask for Help](#)
 - [Discuss AirInv](#)

1.3 AirInv Development

- [Git Repository](#) (Subversion is deprecated)
- [Coding Rules](#)
- [Documentation Rules](#)
- [Test Rules](#)

1.4 External Libraries

- [Boost](#) (C++ STL extensions)
- [Python](#)
- [MySQL client](#)
- [SOI](#) (C++ DB API)

1.5 Support Airlnv

1.6 About Airlnv

Airlnv is a C++ library of airline inventory management classes and functions, mainly targeting simulation purposes. [N](#)

Airlnv makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular the [Boost](#) (*C++ Standard Extensions*) library is used.

The Airlnv library originates from the department of Operational Research and Innovation at [Amadeus](#), Sophia Antipolis, France. Airlnv is released under the terms of the [GNU Lesser General Public License](#) (LGPLv2.1) for you to enjoy.

Airlnv should work on [GNU/Linux](#), [Sun Solaris](#), Microsoft Windows (with [Cygwin](#), [MinGW/MSYS](#), or [Microsoft Visual C++ .NET](#)) and [Mac OS X](#) operating systems.

Note

(N) - The Airlnv library is **NOT** intended, in any way, to be used by airlines for production systems. If you want to report issue, bug or feature request, or if you just want to give feedback, have a look on the right-hand side of this page for the preferred reporting methods. In any case, please do not contact Amadeus directly for any matter related to Airlnv.

2 People

2.1 Project Admins

- Denis Arnaud denis_arnaud@users.sourceforge.net ([N](#))
- Anh Quan Nguyen quannaus@users.sourceforge.net ([N](#))

2.2 Developers

- Anh Quan Nguyen quannaus@users.sourceforge.net ([N](#))
- Denis Arnaud denis_arnaud@users.sourceforge.net ([N](#))
- Son Nguyen Kim snguyenkim@users.sourceforge.net ([N](#))
- Nicolas Bondoux nbondoux@users.sourceforge.net ([N](#))

2.3 Retired Developers

- Patrick Grandjean pgrandjean@users.sourceforge.net ([N](#))
- Ngoc-Thach Hoang hoangngocthach@users.sourceforge.net ([N](#))

2.4 Contributors

- Emmanuel Bastien ebastien@users.sourceforge.net (N)
- Christophe Lacombe ddtoof@users.sourceforge.net (N)

2.5 Distribution Maintainers

- **Fedora/RedHat**: Denis Arnaud denis_arnaud@users.sourceforge.net (N)
- **Debian**: Emmanuel Bastien ebastien@users.sourceforge.net (N)

Note

(N) - **Amadeus** employees.

3 Coding Rules

In the following sections we describe the naming conventions which are used for files, classes, structures, local variables, and global variables.

3.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

3.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

- `int myFunctionName (const int& a, int b)`

3.3 Default Naming Rules for Classes and Structures

Each new word in a class or structure name should always start with a capital letter and the words should be separated with an under-score. Abbreviations are written with capital letters. Examples:

- `MyClassName`
- `MyStructName`

3.4 Default Naming Rules for Files

Files are named after the C++ class names.

Source files are named using `.cpp` suffix, whereas header files end with `.hpp` extension. Examples:

- `FlightDate.hpp`
- `SegmentDate.cpp`

3.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

- default empty constructor
- one or more additional constructor(s) that takes input parameters and initializes the class instance
- setup function, preferably named `'setup'` or `'set_parameters'`

Explicit destructor functions are not required, unless they are needed. It shall not be possible to use any of the other member functions unless the class has been properly initiated with the input parameters.

4 Copyright and License

4.1 GNU LESSER GENERAL PUBLIC LICENSE

4.1.1 Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

4.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of

any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

1. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

1. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

1. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

1. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

1. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components

(compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

1. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

- (a) You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

1. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

1. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

1. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

1. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among

countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

1. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

1. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

4.3.1 NO WARRANTY

1. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
1. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

[Source](#)

5 Documentation Rules

5.1 General Rules

All classes in Airlnv should be properly documented with Doxygen comments in include (.hpp) files. Source (.cpp) files should be documented according to a normal standard for well documented C++ code.

An example of how the interface of a class shall be documented in Airlnv is shown here:

```

/*!
 * \brief Brief description of MyClass here
 *
 * Detailed description of MyClass here. With example code if needed.
 */
class MyClass {
public:
    ///! Default constructor
    MyClass(void) { setup_done = false; }

    /*!
     * \brief Constructor that initializes the class with parameters
     *
     * Detailed description of the constructor here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    MyClass(TYPE1 param1, TYPE2 param2) { setup(param1, param2); }

    /*!
     * \brief Setup function for MyClass
     *
     * Detailed description of the setup function here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    void setup(TYPE1 param1, TYPE2 param2);

    /*!
     * \brief Brief description of memberFunction1
     *

```



```

    * Detailed description of memberFunction1 here if needed
    *
    * \param[in]    param1 Description of \a param1 here
    * \param[in]    param2 Description of \a param2 here
    * \param[in,out] param3 Description of \a param3 here
    * \return Description of the return value here
    */
    TYPE4 memberFunction1(TYPE1 param1, TYPE2 param2, TYPE3 &param3);

private:

    bool _setUpDone;          /*!< Variable that checks if the class is properly
                               initialized with parameters */
    TYPE1 _privateVariable1; /*!< Short description of _privateVariable1 here
    TYPE2 _privateVariable2; /*!< Short description of _privateVariable2 here
};

```

5.2 File Header

All files should start with the following header, which include Doxygen's `\file`, `\brief` and `\author` tags, `$Date$` and `$Revisions$` CVS tags, and a common copyright note:

```

/*!
 * \file
 * \brief Brief description of the file here
 * \author Names of the authors who contributed to this code
 * \date Date
 *
 * Detailed description of the file here if needed.
 *
 * -----
 *
 * AirInv - C++ Airline Inventory Management Library
 *
 * Copyright (C) 2009-2010 (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * -----
 */

```

5.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group `'my_group'`:

```

/*!
 * \defgroup my_group Brief description of the group here
 *
 * Detailed description of the group here
 */

```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```

/*!
 * \brief Brief description of myFunction here
 * \ingroup my_group
 *
 * Detailed description of myFunction here
 *
 * \param[in] param1 Description of \a param1 here
 * \param[in] param2 Description of \a param2 here
 * \return Description of the return value here
 */
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);

```

6 Main features

A short list of the main features of AirInv is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

6.1 Network generation

- Network/graph generation

6.2 Inventory generation

- Inventory generation

6.3 Finding travel solutions

- Matching of travel solutions with user requests

6.4 Distributed inventories

- Inventory independent partitions
- MPI-based distribution

6.5 Other features

- CSV input file parsing
- Memory handling

7 Make a Difference

Do not ask what AirSched can do for you. Ask what you can do for AirSched.

You can help us to develop the AirSched library. There are always a lot of things you can do:

- Start using AirSched
- Tell your friends about AirSched and help them to get started using it
- If you find a bug, report it to us. Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the AirSched discussion forums on SourceForge. If you know the answer to a question, help others to overcome their AirSched problems.
- Help us to improve our algorithms. If you know of a better way (e.g. that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help us to port AirSched to new platforms. If you manage to compile AirSched on a new platform, then tell us how you did it.
- Send us your code. If you have a good AirSched compatible code, which you can release under the LGPLv2.1, and you think it should be included in AirSched, then send it to us.
- Become an AirSched developer. Send us an e-mail and tell what you can do for AirSched.

8 Make a new release

8.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of AirInv using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

8.2 Initialisation

Clone locally the full [Git project](#):

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://airinv.git.sourceforge.net/gitroot/airinv/airinv airinvgit
cd airinvgit
git checkout trunk
```

8.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/airinvgit
git checkout trunk
git checkout -b 0.5.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

```
vi CMakeLists.txt
vi autogen.sh
```

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi airinv.spec
```

8.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/airinvgit
git add -A
git commit -m "[Release 0.5.0] Release of version 0.5.0."
git push
```

8.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/airinvgit
git checkout trunk
vi ChangeLog
vi airinv.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.5.0."
git push
```

8.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/airinvgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airinv-0.5.0 \
  -DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable \
  -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

This will configure, compile and check the package. The output packages will be named, for instance, `airinv-0.5.0.tar.gz` and `airinv-0.5.0.tar.bz2`.

8.7 Generation the RPM packages

Optionally, generate the RPM package (for instance, for [Fedora/RedHat](#)):

```
cd ~/dev/sim/airinvgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airinv-0.5.0 \
  -DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable \
  -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

To perform this step, `rpm-build`, `rpmlint` and `rpmdevtools` have to be available on the system.

```
cp airinv.spec ~/dev/packages/SPECS \
  && cp airinv-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba airinv.spec
rpmlint -i ../SPECS/airinv.spec ../SRPMS/airinv-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/airinv-* ../RPMS/i686/airinv-*
```

8.8 Update distributed change log

Update the `NEWS` and `ChangeLog` files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the [AirInv's Git repository](#).

8.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
make package
```

The output binary package will be named, for instance, `airinv-0.5.0-Linux.tar.bz2`. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

8.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check [SourceForge help page on uploading software](#).

8.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with rsync and SSH:

```
cd ~/dev/sim/airinvgit
git checkout 0.5.0
rsync -aiv doc/html/ doc/latex/refman.pdf joe,airinv@web.sourceforge.net:htdocs/
```

where `-aiv` options mean:

- `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
- `-v`: increase verbosity
- `-i`: output a change-summary for all updates
- Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.

- or use the [SourceForge Shell service](#).

8.12 Make a new post

- submit a new entry in the [SourceForge project-related news feed](#)
- make a new post on the [SourceForge hosted WordPress blog](#)
- and update, if necessary, [Trac tickets](#).

8.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to airinv-announce@lists.sourceforge.net (see <https://lists.sourceforge.net/lists/listinfo/airinv-announce> for the archives)

9 Installation

9.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [Airinv Requirements](#)
- [Basic Installation](#)
- [Compilers and Options](#)
- [Compiling For Multiple Architectures](#)
- [Installation Names](#)
- [Optional Features](#)
- [Particular systems](#)
- [Specifying the System Type](#)
- [Sharing Defaults](#)
- [Defining Variables](#)
- [‘cmake’ Invocation](#)

9.2 Fedora/RedHat Linux distributions

Note that on [Fedora/RedHat](#) Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install airinv-devel airinv-doc
```

RPM packages can also be available on the [SourceForge download site](#).

9.3 Airinv Requirements

Airinv should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:
 - [autoconf](#),
 - [automake](#),
 - [libtool](#),
 - [make](#), version 3.72.1 or later (check version with `'make --version'`)
- [GCC](#) - GNU C++ Compiler (g++), version 4.3.x or later (check version with `'gcc --version'`)
- [Boost](#) - C++ STL extensions, version 1.35 or later (check version with `'grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp'`)
- [MySQL](#) - Database client libraries, version 5.0 or later (check version with `'mysql --version'`)
- [SOCl](#) - C++ database client library wrapper, version 3.0.0 or later (check version with `'soci-config --version'`)

Optionally, you might need a few additional programs: [Doxygen](#), [LaTeX](#), [Dvips](#) and [Ghostscript](#), to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of Airinv.

9.4 Basic Installation

Briefly, the shell commands `'./cmake .. && make install'` should configure, build, and install this package. The following more-detailed instructions are generic; see the `'README'` file for instructions specific to this package. Some packages provide this `'INSTALL'` file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The `'cmake'` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `'Makefile'` in each directory of the package. It may also create one or more `'.h'` files containing system-dependent definitions. Finally, it creates a `'CMakeCache.txt'` cache file that you can refer to in the future to recreate the current configuration, and a file `'CMakeFiles'` containing compiler output (useful mainly for debugging `'cmake'`).

It can also use an optional file (typically called `'config.cache'` and enabled with `'--cache-file=config.cache'` or simply `'-C'`) that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how `'configure'` could check whether to do them, and mail diffs or instructions to the address given in the `'README'` so they can be considered for the

next release. If you are using the cache, and at some point `'config.cache'` contains results you don't want to keep, you may remove or edit it.

The file `<tt>'CMakeLists.txt'</tt>` is used to create the `\c 'Makefile'`

files.

The simplest way to compile this package is:

1. `'cd'` to the directory containing the package's source code and type `'./cmake .'` to configure the package for your system. Running `'cmake'` is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type `'make'` to compile the package.
3. Optionally, type `'make check'` to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type `'make install'` to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the `'make install'` phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing `'make clean'`. To also remove the files that `'configure'` created (so you can compile the package for a different kind of computer), type `'make distclean'`. There is also a `'make maintainer-clean'` target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
6. Often, you can also type `'make uninstall'` to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

9.5 Compilers and Options

Some systems require unusual options for compilation or linking that the `'cmake'` script does not know about. Run `'./cmake --help'` for details on some of the pertinent environment variables.

You can give `'cmake'` initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./cmake CC=c99 CFLAGS=-g LIBS=-lposix
```

See also

[Defining Variables](#) for more details.

9.6 Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU `'make'`. `'cd'` to the directory where you want the object files and executables to go and run the `'configure'` script. `'configure'` automatically checks for the source code in the directory that `'configure'` is in and in `'..'` . This is known as a "VPATH" build.

With a non-GNU 'make', it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types--known as "fat" or "universal" binaries--by specifying multiple '-arch' options to the compiler but only a single '-arch' option to the preprocessor. Like this:

```
./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
           CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
           CPP="gcc -E" CXXCPP="g++ -E"
```

This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the 'lipo' tool if you have problems.

9.7 Installation Names

By default, 'make install' installs the package's commands under '/usr/local/bin', include files under '/usr/local/include', etc. You can specify an installation prefix other than '/usr/local' by giving 'configure' the option '--prefix=PREFIX', where PREFIX must be an absolute file name.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of '\${prefix}', so that specifying just '--prefix' will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to 'configure'; however, many packages provide one or both of the following shortcuts of passing variable assignments to the 'make install' command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, 'make install prefix=/alternate/directory' will choose an alternate location for all directory configuration variables that were expressed in terms of '\${prefix}'. Any directories that were specified during 'configure', but not in terms of '\${prefix}', must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the GNU Coding Standards, and does not work on platforms that

have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of `'${prefix}'` at `'configure'` time.

9.8 Optional Features

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving `'cmake'` the option `'--program-prefix=PREFIX'` or `'--program-suffix=SUFFIX'`.

Some packages pay attention to `'--enable-FEATURE'` options to `'configure'`, where `FEATURE` indicates an optional part of the package. They may also pay attention to `'--with-PACKAGE'` options, where `PACKAGE` is something like `'gnu-as'` or `'x'` (for the X Window System). The `'README'` should mention any `'--enable-'` and `'--with-'` options that the package recognizes.

For packages that use the X Window System, `'configure'` can usually find the X include and library files automatically, but if it doesn't, you can use the `'configure'` options `'--x-includes=DIR'` and `'--x-libraries=DIR'` to specify their locations.

Some packages offer the ability to configure how verbose the execution of `'make'` will be. For these packages, running `'./configure --enable-silent-rules'` sets the default to minimal output, which can be overridden with `'make V=1'`; while running `'./configure --disable-silent-rules'` sets the default to verbose, which can be overridden with `'make V=0'`.

9.9 Particular systems

On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn't work, install pre-built binaries of GCC for HP-UX.

On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its `<wchar.h>` header file. The option `'-nodtk'` can be used as a workaround. If GNU CC is not installed, it is therefore recommended to try

```
./configure CC="cc"
```

and if that doesn't work, try

```
./configure CC="cc -nodtk"
```

On Solaris, don't put `'/usr/ucb'` early in your `'PATH'`. This directory contains several dysfunctional programs; working variants of these programs are available in `'/usr/bin'`. So, if you need `'/usr/ucb'` in your `'PATH'`, put it *after* `'/usr/bin'`.

On Haiku, software installed for all users goes in `'/boot/common'`, not `'/usr/local'`. It is recommended to use the following options:

```
./cmake -DCMAKE_INSTALL_PREFIX=/boot/common
```

9.10 Specifying the System Type

There may be some features `'configure'` cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the *same* architectures, `'configure'` can figure that out, but if it prints a message saying it cannot guess the machine type, give it the `'--build=TYPE'` option. `TYPE` can either be a short name for the system type, such as `'sun4'`, or a canonical name which has the form `CPU-COMPANY-SYSTEM`

where `SYSTEM` can have one of these forms:

- OS
- KERNEL-OS

See the file `'config.sub'` for the possible values of each field. If `'config.sub'` isn't included in this package, then this package doesn't need to know the machine type.

If you are *building* compiler tools for cross-compiling, you should use the option `'--target=TYPE'` to select the type of system they will produce code for.

If you want to use a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e., that on which the generated programs will eventually be run) with `'--host=TYPE'`.

9.11 Sharing Defaults

If you want to set default values for `'configure'` scripts to share, you can create a site shell script called `'config.site'` that gives default values for variables like `'CC'`, `'cache_file'`, and `'prefix'`. `'configure'` looks for `'PREFIX/share/config.site'` if it exists, then `'PREFIX/etc/config.site'` if it exists. Or, you can set the `'CONFIG_SITE'` environment variable to the location of the site script. A warning: not all `'configure'` scripts look for a site script.

9.12 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to `'configure'`. However, some packages may run `configure` again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the `'configure'` command line, using `'VAR=value'`. For example:

```
./configure CC=/usr/local2/bin/gcc
```

causes the specified `'gcc'` to be used as the C compiler (unless it is overridden in the site shell script).

Unfortunately, this technique does not work for `'CONFIG_SHELL'` due to an Autoconf bug. Until the bug is fixed you can use this workaround:

```
CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash
```

9.13 'cmake' Invocation

`'cmake'` recognizes the following options to control how it operates.

- '--help', '-h' print a summary of all of the options to 'cmake', and exit.
- '--help=short', '--help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '--version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '--cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '--config-cache', '-C' alias for '--cache-file=config.cache'.
- '--quiet', '--silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '--srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '--prefix=DIR' use DIR as the installation prefix.

See also

[Installation Names](#) for more details, including other options available for fine-tuning the installation locations.

- '--no-create', '-n' run the configure checks, but stop before creating any output files.

'cmake' also accepts some other, not widely useful, options. Run 'cmake' --help' for more details.

The 'cmake' script produces an output like this:

```
export LIBSUFFIX_4_CMAKE="-DLIB_SUFFIX=64"
export INSTALL_BASEDIR=/home/user/dev/deliveries
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_BASEDIR}/airinv-0.5.0 \
  -DWITH_STDAIR_PREFIX=${INSTALL_BASEDIR}/stdair-stable \
  -DWITH_AIRAC_PREFIX=${INSTALL_BASEDIR}/airrac-stable \
  -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/rmol-stable \
  -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ${LIBSUFFIX_4_CMAKE} ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: 0ee8dcc3e3dd1d442c4054fbfa4cacc1182e6a trunk
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   regex
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
```

```

-- python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires Readline without specifying any version
-- Found Readline: /usr/include
-- Found Readline version: 6.2
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.37
-- Found StdAir version: 0.38.0
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'airraclib' to CXX
-- Had to set the linker language for 'rmollib' to CXX
-- Had to set the linker language for 'airinvlib' to CXX
-- Test 'InventoryTestSuite' to be built with 'InventoryTestSuite.cpp'
--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : airinv
-- PACKAGE_PRETTY_NAME ..... : AirInv
-- PACKAGE ..... : airinv
-- PACKAGE_NAME ..... : AIRINV
-- PACKAGE_BRIEF ..... : C++ Simulated Airline Inventory Management System library
-- PACKAGE_VERSION ..... : 0.5.0
-- GENERIC_LIB_VERSION ..... : 0.5.0
-- GENERIC_LIB_SOVERSION ..... : 0.5
--
-- -----
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : airrac;rmol;airinv
-- Libraries to build/install ..... : airraclib;rmollib;airinvlib
-- Binaries to build/install ..... : airrac;rmol;airinv_parseInventory;airinv
-- Modules to test ..... : airinv
-- Binaries to test ..... : InventoryTestSuitetst
--
-- * Module ..... : airrac
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : airraclib
--   + Executables to build/install : airrac
--   + Tests to perform ..... :
-- * Module ..... : rmol
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib
--   + Libraries to build/install . : rmollib
--   + Executables to build/install : rmol
--   + Tests to perform ..... :
-- * Module ..... : airinv
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib;rmollib
--   + Libraries to build/install . : airinvlib
--   + Executables to build/install : airinv_parseInventory;airinv
--   + Tests to perform ..... : InventoryTestSuitetst
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -Wall -Werror
-- * BUILD_FLAGS ..... :

```

```

-- * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/dan/dev/sim/airinv/airinvgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/dan/dev/deliveries/airinv-0.5.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ..... : 1.7.4
--   - DOXYGEN_EXECUTABLE ..... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ..... : /usr/bin
--
-----
-- --- Installation Configuration ---
-----
-- INSTALL_LIB_DIR ..... : /home/dan/dev/deliveries/airinv-0.5.0/lib64
-- INSTALL_BIN_DIR ..... : /home/dan/dev/deliveries/airinv-0.5.0/bin
-- INSTALL_INCLUDE_DIR ..... : /home/dan/dev/deliveries/airinv-0.5.0/include
-- INSTALL_DATA_DIR ..... : /home/dan/dev/deliveries/airinv-0.5.0/share
-- INSTALL_SAMPLE_DIR ..... : /home/dan/dev/deliveries/airinv-0.5.0/share/airinv/samples
-- INSTALL_DOC ..... : ON
--
-----
-- --- Packaging Configuration ---
-----
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 0.5.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/dan/dev/sim/airinv/airinvgithub/README
-- CPACK_RESOURCE_FILE_LICENSE ... : /home/dan/dev/sim/airinv/airinvgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : airinv-0.5.0
--
-----
-- --- External libraries ---
-----
--
-- * Boost:
--   - Boost_VERSION ..... : 104600
--   - Boost_LIB_VERSION ..... : 1_46
--   - Boost_HUMAN_VERSION ..... : 1.46.0
--   - Boost_INCLUDE_DIRS ..... : /usr/include
--   - Boost required components .. : regex;program_options;date_time;iostreams;serialization;filesystem;unit_
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_rege
--
-- * Readline:
--   - READLINE_VERSION ..... : 6.2
--   - READLINE_INCLUDE_DIR ..... : /usr/include
--   - READLINE_LIBRARY ..... : /usr/lib64/libreadline.so
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ..... : 0.38.0
--   - STDAIR_BINARY_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/bin
--   - STDAIR_EXECUTABLES ..... : stdair
--   - STDAIR_LIBRARY_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/lib64
--   - STDAIR_LIBRARIES ..... : stdairlib;stdairuiclib
--   - STDAIR_INCLUDE_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/include
--   - STDAIR_SAMPLE_DIR ..... : /home/dan/dev/deliveries/stdair-0.38.0/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====

```

```
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/dan/dev/sim/airinv/airinvgithub/build
```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```
[ 0%] Built target hdr_cfg_airinv
[ 0%] Built target hdr_cfg_airrac
[ 13%] Built target airraclib
[ 13%] Built target hdr_cfg_rmol
[ 38%] Built target rmolib
[ 98%] Built target airinvlib
[100%] Built target InventoryTestSuitetst
Scanning dependencies of target check_airinvtst
Test project /home/dan/dev/sim/airinv/airinvgithub/build/test/airinv
  Start 1: InventoryTestSuitetst
1/1 Test #1: InventoryTestSuitetst ..... Passed    0.08 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.35 sec
[100%] Built target check_airinvtst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a [bug-report](#).

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/airinvgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with Airinv

10.1 Table of Contents

- [Introduction](#)
- [Dependencies](#)
- [Using the pkg-config command](#)
- [Using the airinv-config script](#)
- [M4 macro for the GNU Autotools](#)
- [Using Airinv with dynamic linking](#)

10.2 Introduction

There are two convenient methods of linking your programs with the Airinv library. The first one employs the `'pkg-config'` command (see <http://pkgconfig.freedesktop.org/>), whereas the second one uses `'airinv-config'` script. These methods are shortly described below.

10.3 Dependencies

The Airinv library depends on several other C++ components.

10.3.1 StdAir

Among them, as for now, only StdAir has been packaged. The support for StdAir is taken in charge by a dedicated M4 macro file (namely, `'stdair.m4'`), from the configuration script (generated thanks to `'configure.ac'`).

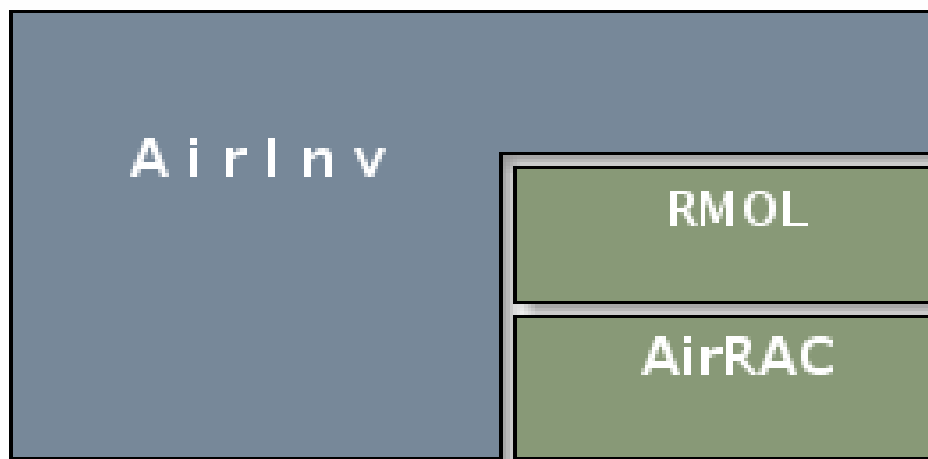


Figure 1: Airinv Dependencies

10.4 Using the pkg-config command

`'pkg-config'` is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the `'pkg-config'` is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an Airinv based program `'my_prog.cpp'`, you should use the following command:

```
g++ `pkg-config --cflags airinv` -o my_prog my_prog.cpp `pkg-config --libs airinv`
```

For more information see the `'pkg-config'` man pages.

10.5 Using the airinv-config script

Airinv provides a shell script called `airinv-config`, which is installed by default in ``$prefix/bin'` (``/usr/local/bin'`) directory. It can be used to simplify compilation and linking of Airinv based programs. The usage of this script is quite similar to the usage of the ``pkg-config'` command.

Assuming that you need to compile the program ``my_prog.cpp'` you can now do that with the following command:

```
g++ `airinv-config --cflags` -o my_prog_opt my_prog.cpp `airinv-config --libs`
```

A list of ``airinv-config'` options can be obtained by typing:

```
airinv-config --help
```

If the ``airinv-config'` command is not found by your shell, you should add its location ``$prefix/bin'` to the `PATH` environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

10.6 M4 macro for the GNU Autotools

A M4 macro file is delivered with Airinv, namely ``airinv.m4'`, which can be found in, e.g., ``/usr/share/aclocal'`. When used by a ``configure'` script, thanks to the ``AM_PATH_Airinv'` macro (specified in the M4 macro file), the following Makefile variables are then defined:

- ``Airinv_VERSION'` (e.g., defined to 0.23.0)
- ``Airinv_CFLAGS'` (e.g., defined to ``-I${prefix}/include'`)
- ``Airinv_LIBS'` (e.g., defined to ``-L${prefix}/lib -lairinv'`)

10.7 Using Airinv with dynamic linking

When using static linking some of the library routines in Airinv are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared Airinv library file during your program execution. If you install the Airinv library using a non-standard prefix, the ``LD_LIBRARY_PATH'` environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<Airinv installation prefix>/lib:$LD_LIBRARY_PATH
```

11 Test Rules

This section describes rules how the functionality of the IT++ library should be verified. In the ``tests'` subdirectory test files are provided. All functionality should be tested using these test files.

11.1 The Test File

Each new IT++ module/class should be accompanied with a test file. The test file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called modules. The test file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test files should be maintained using version control and updated whenever new functionality is added to the IT++ library.

The test file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test file should be placed in the ``tests'` subdirectory and should have a name ending with ``_test.cpp'`.

11.2 The Reference File

Consider a test file named `'module_test.cpp'`. A reference file named `'module_test.ref'` should accompany the test file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test file.

11.3 Testing IT++ Library

One can compile and execute all test programs from `'tests'` subdirectory by typing

```
% make check
```

after successful compilation of the IT++ library.

12 Users Guide

12.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
 - [Get the AirInv library](#)
 - [Build the AirInv project](#)
 - [Build and Run the Tests](#)
 - [Install the AirInv Project \(Binaries, Documentation\)](#)
- [Input file of AirInv Project](#)
- [The schedule BOM Tree](#)
 - [Build of the schedule BOM tree](#)
 - [Display of the schedule BOM tree](#)
- [Exploring the Predefined BOM Tree](#)
 - [Airline Network BOM Tree](#)
 - [Airline Schedule BOM Tree](#)
- [Extending the BOM Tree](#)
- [The travel solution calculation procedure](#)

12.2 Introduction

The `AirInv` library contains classes for airline business management. This document does not cover all the aspects of the `AirInv` library. It does however explain the most important things you need to know in order to start using `AirInv`.

12.3 Get Started

12.3.1 Get the AirInv library

Clone locally the full [Git project](#):

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://airinv.git.sourceforge.net/gitroot/airinv/airinv airinvgit
cd airinvgit
git checkout trunk
```

12.3.2 Build the AirInv project

Link with StdAir, create the distribution package (say, 0.5.0) and compile using the following commands:

```
cd ~/dev/sim/airinvgit
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=~/dev/deliveries/airinv-0.5.0 \
      -DWITH_STDAIR_PREFIX=~/dev/deliveries/stdair-stable \
      -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make
```

12.3.3 Build and Run the Tests

After building the AirInv project, the following commands run the tests:

```
cd ~/dev/sim/airinvgit
cd build
make check
```

As a result, you should obtain a similar report:

```
[ 0%] Built target hdr_cfg_airinv
[ 96%] Built target airinvlib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airinvtst
Test project /home/dan/dev/sim/airinv/airinvgithub/build/test/airinv
  Start 1: AirlineScheduleTestSuitetst
1/1 Test #1: AirlineScheduleTestSuitetst ..... Passed    0.15 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.40 sec
[100%] Built target check_airinvtst
Scanning dependencies of target check
[100%] Built target check
```

12.3.4 Install the AirInv Project (Binaries, Documentation)

After the step [Build the AirInv project](#), to install the library and its header files, type:

```
cd ~/dev/sim/airinvgit
cd build
make install
```

You can check that the executables and other required files have been copied into the given final directory:

```
cd ~/dev/deliveries/airinv-0.5.0
```

To generate the AirInv project documentation, the commands are:

```
cd ~/dev/sim/airinvgit
cd build
make doc
```

The AirInv project documentation is available in the following formats: HTML, LaTeX. Those documents are available in a subdirectory:

```
cd ~/dev/sim/airinvgit
cd build
cd doc
```

12.4 Input file of AirInv Project

The schedule input file structure should look like the following sample:

Each line, beyond the header, represents a schedule entry, i.e., the specification of a given flight-period (see [AIR-INV::FlightPeriodStruct](#)). The fields are as follows:

- Flights section
 - AirlineCode (e.g., BA)
 - FlightNumber (e.g., 9)
 - Start of the flight departure period (e.g., 2007-04-20)
 - End of the flight departure period (e.g., 2007-06-30)
 - Day-Of-the-Week for the flight departure period (DOW) (e.g., 0000011)
 - Leg section
 - Segment section
- Leg section
 - BoardPoint (e.g., LHR)
 - OffPoint (e.g., BKK)
 - BoardTime (e.g., 22:00)
 - ArrivalTime (e.g., 15:15)
 - ArrivalDateOffSet (e.g., +1)
 - ElapsedTime (e.g., 11:15)
 - Leg-cabin section
- Leg-cabin section
 - Cabin code (e.g., F, J, W or Y)
 - Capacity (e.g., respectively 5, 12, 20 or 300)
- Segment section
 - Specificity flag:
 - * 0 means that all the segments behave the same way, i.e., have got the same dressing (distribution and order of the booking classes per cabin)
 - * 1 means that each segment behave differently. The full specification of each of those segments must therefore be given.
 - Segment-cabin section
 - Fare family section

- Segment-cabin section
 - Cabin code (e.g., F, J, W or Y)
 - List of (one-letter-code) booking classes for the cabin (e.g, respectively FA, JC DI, WT or YBHKMLSQ)
- Fare family section
 - Fare family code (e.g., 1)
 - List of (one-letter-code) booking classes for the fare family (e.g, respectively FA, JC DI, WT or YBHKMLSQ)

Some fare input examples (including the example above named `schedule03.csv`) are given in the `StdAir` project.

12.5 The schedule BOM Tree

The schedule-related Business Object Model (BOM) tree is a structure allowing to store all the `AIRINV::FlightPeriodStruct` objects of the simulation. That is why parsing an input file, containing the specification for all the flight-periods, is more convenient (

See also

the previous section [Input file of AirInv Project](#)).

As it may be time consuming, and it for sure requires some know-how, to first build such a schedule input file, a small sample BOM tree is provided by default when needed.

12.5.1 Build of the schedule BOM tree

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated (during the instantiation of the `AIRINV::AIRINV_Service` object).

The corresponding type (class) `stdair::BomRoot` is defined in the `StdAir` library.

Then, the BOM root can be either constructed thanks to the `AIRINV::AIRINV_Service::buildSampleBom()` method:

```
void buildSampleBom();
```

or can be constructed using the schedule input file described above thanks to the `AIRINV::AIRINV_Service::parseAndLoad` (`const stdair::Filename_T&`) method:

```
void parseAndLoad (const stdair::Filename_T& iInventoryFilename);
```

12.5.2 Display of the schedule BOM tree

Note

That feature (of BOM tree display) has not been implemented yet. Do not hesitate to [open a ticket](#) if you would like to have it implemented more quickly.

The schedule BOM tree can be displayed as done in the `batches::airinv.cpp` program:

When the default BOM tree is used (-b/--builtin option of the main program `airinv.cpp`), the schedule BOM tree display (for now, corresponding to `schedule01.csv` parsed by `AIRINV::parseInventory`) should look like:

```
=====
BomRoot:  -- ROOT  --
=====
+++++
Inventory: SQ
+++++
*****
FlightDate: SQ11, 2010-Jan-15
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-15, SIN-BKK, 2010-Jan-15, 08:20:00, 2010-Jan-15, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 2, 298
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, 0, 0, 0, 2, 298, 0,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, 0, 0, 0, 2, 298, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, Y, 300 (0), 0, 0, 0, 2, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-16
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-16, SIN-BKK, 2010-Jan-16, 08:20:00, 2010-Jan-16, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 1.83244e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
```

```

      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-17
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-17, SIN-BKK, 2010-Jan-17, 08:20:00, 2010-Jan-17, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 1.58896e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-18, SIN-BKK, 2010-Jan-18, 08:20:00, 2010-Jan-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****

```

```

*****
*****
FlightDate: SQ11, 2010-Jan-19
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-19, SIN-BKK, 2010-Jan-19, 08:20:00, 2010-Jan-19, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-20
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-20, SIN-BKK, 2010-Jan-20, 08:20:00, 2010-Jan-20, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-21
*****
*****

```

```
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-21, SIN-BKK, 2010-Jan-21, 08:20:00, 2010-Jan-21, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-22
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-22, SIN-BKK, 2010-Jan-22, 08:20:00, 2010-Jan-22, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-23, SIN-BKK, 2010-Jan-23, 08:20:00, 2010-Jan-23, 11:00:00, 07:40:
```



```
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 6.64029e-
319, 0, 300, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-24
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-24, SIN-BKK, 2010-Jan-24, 08:20:00, 2010-Jan-24, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-25
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-25, SIN-BKK, 2010-Jan-25, 08:20:00, 2010-Jan-25, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
```

```

Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-26, SIN-BKK, 2010-Jan-26, 08:20:00, 2010-Jan-26, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-27
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Jan-27, SIN-BKK, 2010-Jan-27, 08:20:00, 2010-Jan-27, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****

```

```
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, 0, 0, 0, 0, 300, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-28
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-28, SIN-BKK, 2010-Jan-28, 08:20:00, 2010-Jan-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-29
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-29, SIN-BKK, 2010-Jan-29, 08:20:00, 2010-Jan-29, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
```

```

*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-30
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-30, SIN-BKK, 2010-Jan-30, 08:20:00, 2010-Jan-30, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Jan-31
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-31, SIN-BKK, 2010-Jan-31, 08:20:00, 2010-Jan-31, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, 0, 0, 0, 0, 300, 0,

```

```

SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, 0, 0, 0, 0, 300, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-01
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, 08:20:00, 2010-Feb-01, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-02
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, 08:20:00, 2010-Feb-02, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----

```

```
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-03
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-03, SIN-BKK, 2010-Feb-03, 08:20:00, 2010-Feb-03, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-04
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-04, SIN-BKK, 2010-Feb-04, 08:20:00, 2010-Feb-04, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
```

```

0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-05
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-05, SIN-BKK, 2010-Feb-05, 08:20:00, 2010-Feb-05, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-06
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-06, SIN-BKK, 2010-Feb-06, 08:20:00, 2010-Feb-06, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-07
*****
*****

```

```
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-07, SIN-BKK, 2010-Feb-07, 08:20:00, 2010-Feb-07, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-08
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-08, SIN-BKK, 2010-Feb-08, 08:20:00, 2010-Feb-08, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-09
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
```



```
SQL1 2010-Feb-09, SIN-BKK, 2010-Feb-09, 08:20:00, 2010-Feb-09, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL1 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL1 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, 0, 0, 0, 0, 300, 0,
SQL1 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL1 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL1 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL1, 2010-Feb-10
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL1 2010-Feb-10, SIN-BKK, 2010-Feb-10, 08:20:00, 2010-Feb-10, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL1 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL1 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, 0, 0, 0, 0, 300, 0,
SQL1 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL1 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL1 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL1, 2010-Feb-11
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL1 2010-Feb-11, SIN-BKK, 2010-Feb-11, 08:20:00, 2010-Feb-11, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
```

```
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-12
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-12, SIN-BKK, 2010-Feb-12, 08:20:00, 2010-Feb-12, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-13
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-13, SIN-BKK, 2010-Feb-13, 08:20:00, 2010-Feb-13, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
```

```

*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-14
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-14, SIN-BKK, 2010-Feb-14, 08:20:00, 2010-Feb-14, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-15
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-15, SIN-BKK, 2010-Feb-15, 08:20:00, 2010-Feb-15, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,

```

```
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-16
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-16, SIN-BKK, 2010-Feb-16, 08:20:00, 2010-Feb-16, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-17
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ11 2010-Feb-17, SIN-BKK, 2010-Feb-17, 08:20:00, 2010-Feb-17, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
```

```

SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, 0, 0, 0, 0, 300, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-18, SIN-BKK, 2010-Feb-18, 08:20:00, 2010-Feb-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-19
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-19, SIN-BKK, 2010-Feb-19, 08:20:00, 2010-Feb-19, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:

```

```

-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-20
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
    Elapsed, Distance, Capacity,
SQ11 2010-Feb-20, SIN-BKK, 2010-Feb-20, 08:20:00, 2010-Feb-20, 11:00:00, 07:40:
    00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
    CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
    , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-21
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
    Elapsed, Distance, Capacity,
SQ11 2010-Feb-21, SIN-BKK, 2010-Feb-21, 08:20:00, 2010-Feb-21, 11:00:00, 07:40:
    00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
    CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
    , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,

```

```

SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-22
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-22, SIN-BKK, 2010-Feb-22, 08:20:00, 2010-Feb-22, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-23, SIN-BKK, 2010-Feb-23, 08:20:00, 2010-Feb-23, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-24

```

```
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-24, SIN-BKK, 2010-Feb-24, 08:20:00, 2010-Feb-24, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-25
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ11 2010-Feb-25, SIN-BKK, 2010-Feb-25, 08:20:00, 2010-Feb-25, 11:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
```



```
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-26, SIN-BKK, 2010-Feb-26, 08:20:00, 2010-Feb-26, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-27
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-27, SIN-BKK, 2010-Feb-27, 08:20:00, 2010-Feb-27, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-28
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-28, SIN-BKK, 2010-Feb-28, 08:20:00, 2010-Feb-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
```

```

LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-15
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-15, SIN-HND, 2010-Jan-15, 09:20:00, 2010-Jan-15, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 200, 200, 2.082e+121, 5.53287e-48, 5.
20268e-90, 0, 1.31346e-47, 1.05119e-153, 2.78986e+179, 0, 200, 9, 3.66962e-62, 1
.0854e-71, 6.74783e-67, 6.9835e-77, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, Y13856, 200 (0), 0, 0, 0, 0, 0 (0)
, 0, 0, 0, 0, 0, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-16
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-16, SIN-HND, 2010-Jan-16, 09:20:00, 2010-Jan-16, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,

```

```
SQL12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.63638e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL12, 2010-Jan-17
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL12 2010-Jan-17, SIN-HND, 2010-Jan-17, 09:20:00, 2010-Jan-17, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.39291e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL12, 2010-Jan-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL12 2010-Jan-18, SIN-HND, 2010-Jan-18, 09:20:00, 2010-Jan-18, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.14469e-319, 0, 0, 0, 0,
*****
*****
Buckets:
```

```
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Jan-19
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-19, SIN-HND, 2010-Jan-19, 09:20:00, 2010-Jan-19, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Jan-20
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-20, SIN-HND, 2010-Jan-20, 09:20:00, 2010-Jan-20, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
```

```

-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Jan-21
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ12 2010-Jan-21, SIN-HND, 2010-Jan-21, 09:20:00, 2010-Jan-21, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Jan-22
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ12 2010-Jan-22, SIN-HND, 2010-Jan-22, 09:20:00, 2010-Jan-22, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 2, 0, 0, 0, 0, 200, 0,
*****

```

```
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
    Elapsed, Distance, Capacity,
SQ12 2010-Jan-23, SIN-HND, 2010-Jan-23, 09:20:00, 2010-Jan-23, 12:00:00, 07:40:
    00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
    CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
    , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
    0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-24
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
    Elapsed, Distance, Capacity,
SQ12 2010-Jan-24, SIN-HND, 2010-Jan-24, 09:20:00, 2010-Jan-24, 12:00:00, 07:40:
    00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
    CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
    , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
    GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
```

```
SQL12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL12, 2010-Jan-25
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL12 2010-Jan-25, SIN-HND, 2010-Jan-25, 09:20:00, 2010-Jan-25, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL12, 2010-Jan-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL12 2010-Jan-26, SIN-HND, 2010-Jan-26, 09:20:00, 2010-Jan-26, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQL12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
```

```

*****
FlightDate: SQ12, 2010-Jan-27
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-27, SIN-HND, 2010-Jan-27, 09:20:00, 2010-Jan-27, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-28
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-28, SIN-HND, 2010-Jan-28, 09:20:00, 2010-Jan-28, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-29
*****
*****
Leg-Dates:

```



```

-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-29, SIN-HND, 2010-Jan-29, 09:20:00, 2010-Jan-29, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-30
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-30, SIN-HND, 2010-Jan-30, 09:20:00, 2010-Jan-30, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-31
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Jan-31, SIN-HND, 2010-Jan-31, 09:20:00, 2010-Jan-31, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,

```

```
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-01
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-01, SIN-HND, 2010-Feb-01, 09:20:00, 2010-Feb-01, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 200, 200, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-02
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-02, SIN-HND, 2010-Feb-02, 09:20:00, 2010-Feb-02, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
```

```
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-03
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-03, SIN-HND, 2010-Feb-03, 09:20:00, 2010-Feb-03, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-04
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-04, SIN-HND, 2010-Feb-04, 09:20:00, 2010-Feb-04, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
```

```
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-05
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ12 2010-Feb-05, SIN-HND, 2010-Feb-05, 09:20:00, 2010-Feb-05, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-06
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ12 2010-Feb-06, SIN-HND, 2010-Feb-06, 09:20:00, 2010-Feb-06, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
```

```
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-07
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-07, SIN-HND, 2010-Feb-07, 09:20:00, 2010-Feb-07, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-08
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-08, SIN-HND, 2010-Feb-08, 09:20:00, 2010-Feb-08, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, 0, 0, 0, 0, 200, 0,
```

```

*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-09
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-09, SIN-HND, 2010-Feb-09, 09:20:00, 2010-Feb-09, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-10
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-10, SIN-HND, 2010-Feb-10, 09:20:00, 2010-Feb-10, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,

```

```

      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-11
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-11, SIN-HND, 2010-Feb-11, 09:20:00, 2010-Feb-11, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-12
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-12, SIN-HND, 2010-Feb-12, 09:20:00, 2010-Feb-12, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,

```

```
*****
*****
FlightDate: SQ12, 2010-Feb-13
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-13, SIN-HND, 2010-Feb-13, 09:20:00, 2010-Feb-13, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-14
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-14, SIN-HND, 2010-Feb-14, 09:20:00, 2010-Feb-14, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-15
*****
*****
```



```
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-15, SIN-HND, 2010-Feb-15, 09:20:00, 2010-Feb-15, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, Y, 200 (0), 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-16
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-16, SIN-HND, 2010-Feb-16, 09:20:00, 2010-Feb-16, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-17
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-17, SIN-HND, 2010-Feb-17, 09:20:00, 2010-Feb-17, 12:00:00, 07:40:
```

```
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-18, SIN-HND, 2010-Feb-18, 09:20:00, 2010-Feb-18, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-19
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-19, SIN-HND, 2010-Feb-19, 09:20:00, 2010-Feb-19, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
```

```
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-20
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-20, SIN-HND, 2010-Feb-20, 09:20:00, 2010-Feb-20, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-21
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQ12 2010-Feb-21, SIN-HND, 2010-Feb-21, 09:20:00, 2010-Feb-21, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
```

```
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-22
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-22, SIN-HND, 2010-Feb-22, 09:20:00, 2010-Feb-22, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-23, SIN-HND, 2010-Feb-23, 09:20:00, 2010-Feb-23, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
```

```

*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Feb-24
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-24, SIN-HND, 2010-Feb-24, 09:20:00, 2010-Feb-24, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
*****
FlightDate: SQ12, 2010-Feb-25
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-25, SIN-HND, 2010-Feb-25, 09:20:00, 2010-Feb-25, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, 0, 0, 0, 0, 200, 0,

```

```
SQL2 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, 0, 0, 0, 0, 200, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL2 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQL2 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL2, 2010-Feb-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQL2 2010-Feb-26, SIN-HND, 2010-Feb-26, 09:20:00, 2010-Feb-26, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL2 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQL2 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
SQL2 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
  0, 0, 0, 0, 0,
*****
*****
FlightDate: SQL2, 2010-Feb-27
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQL2 2010-Feb-27, SIN-HND, 2010-Feb-27, 09:20:00, 2010-Feb-27, 12:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
  , 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL2 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
```

```
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-28
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQ12 2010-Feb-28, SIN-HND, 2010-Feb-28, 09:20:00, 2010-Feb-28, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
  GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
```

12.6 Exploring the Predefined BOM Tree

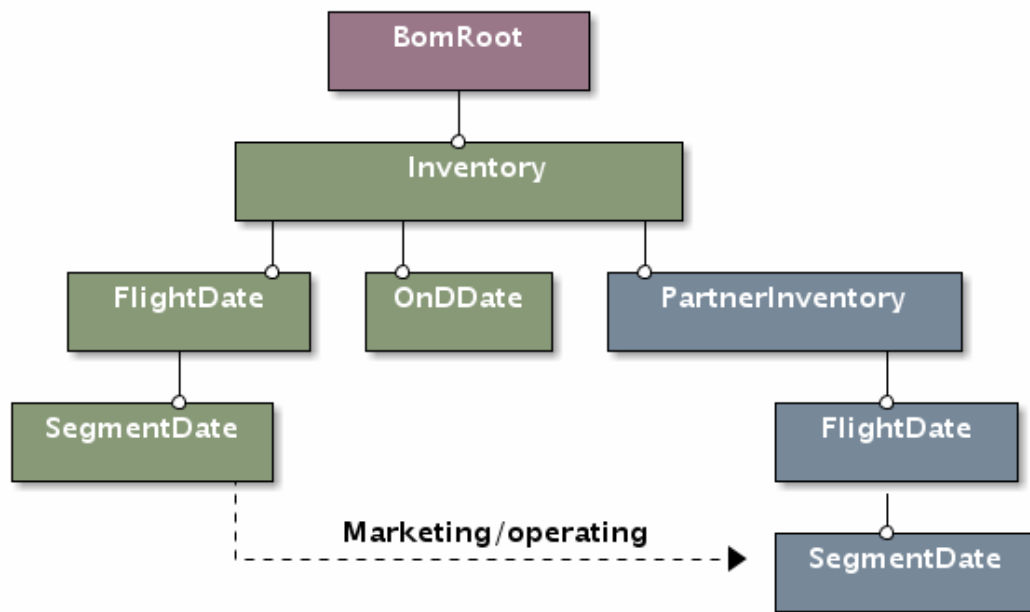


Figure 2: AirInv BOM tree

`AirInv` predefines a BOM (Business Object Model) tree specific to the airline IT arena.

12.6.1 Airline Network BOM Tree

- `AIRINV::ReachableUniverse`
- `AIRINV::OriginDestinationSet`
- `AIRINV::SegmentPathPeriod`

12.6.2 Airline Schedule BOM Tree

- `stdair::Inventory`
- `stdair::FlightPeriod`
- `stdair::SegmentPeriod`
- `stdair::OnDPeriod`

12.7 Extending the BOM Tree

12.8 The travel solution calculation procedure

The project `AirInv` aims at calculating a list of **travel solutions** for every incoming **booking request**.

13 Supported Systems

13.1 Table of Contents

- [Introduction](#)
- [.1 AirInv 0.1.x.1](#)
 - [Linux Systems](#)
 - * [Fedora Core 4 with ATLAS](#)
 - * [Gentoo Linux with ACML](#)
 - * [Gentoo Linux with ATLAS](#)
 - * [Gentoo Linux with MKL](#)
 - * [Gentoo Linux with NetLib's BLAS and LAPACK](#)
 - * [Red Hat Enterprise Linux with AirInv External](#)
 - * [SUSE Linux 10.0 with NetLib's BLAS and LAPACK](#)
 - * [SUSE Linux 10.0 with MKL](#)
 - [Windows Systems](#)
 - * [Microsoft Windows XP with Cygwin](#)
 - * [Microsoft Windows XP with Cygwin and ATLAS](#)
 - * [Microsoft Windows XP with Cygwin and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and AirInv External](#)
 - * [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
 - [Unix Systems](#)
 - * [SunOS 5.9 with AirInv External](#)
- [AirInv 3.9.1](#)
- [AirInv 3.9.0](#)
- [AirInv 3.8.1](#)

13.2 Introduction

This page is intended to provide a list of AirInv supported systems, i.e. the systems on which configuration, installation and testing process of the AirInv library has been successful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the AirInv library on a system not mentioned below, please let us know, so we could update this database.

14 AirInv Supported Systems (Previous Releases)

14.1 AirInv 3.9.1

14.2 AirInv 3.9.0

14.3 AirInv 3.8.1

15 Tutorials

15.1 Table of Contents

- [Preparing the AirSched Project for Development](#)
- [Your first networkBuilde](#)
 - [Summary of the different steps](#)
 - [Result of the Batch Program](#)
- [Network building with an input file](#)
 - [How to build a network input file?](#)
 - [Building the BOM tree with an input file](#)
 - [Result of the Batch Program](#)

15.2 Preparing the AirSched Project for Development

The source code for these examples can be found in the `batches` and `test/airsched` directories. They are compiled along with the rest of the `AirSched` project. See the [Users Guide](#) for more details on how to build the `AirSched` project.

15.3 Your first networkBuilde

15.3.1 Summary of the different steps

All the steps below can be found in the same order in the batch `AirSched.cpp` program.

First, we instantiate the `AIRSCHEM_Service` object:

Then, we construct a default sample list of travel solutions and a default booking request (as mentioned in `ug_procedure_bookingrequest` and `ug_procedure_travelsolution` parts):

For basic use, the default BOM tree can be built using:

The main step is the network building (see [The travel solution calculation procedure](#)):

15.3.2 Result of the Batch Program

When the `AirSched.cpp` program is run (with the `-b` option), the log output file should look like:

What is interesting is to compare the travel solution list (here reduced to a single travel solution) displayed before:

and after the network building:

Between the two groups of dashes, we can see that a network option structure has been added by the network builder: the price is 450 EUR for the Y class, the ticket is refundable but there are exchange fees and the customer must stay over on Saturday night.

Let's return to our default BOM tree display: the only network rule stored was a match for the travel solution into consideration (same origin airport, same destination airport, flight date included in the network rule date range, same airline "BA", ...).

By looking at the network rule trip type "RT", we can guess we face a round trip network: that means the price given in the default bom tree construction in `stdair::CmdBomManager.hpp` has been divided by 2 because we are considering either an inbound trip or an outbound one.

15.4 Network building with an input file

15.4.1 How to build a network input file?

The objective here is to build a network input file to network build the default travel solution list built using:

This travel solution list, reduced to a singleton, can be displayed as done before:

We deduce:

- we need a network rule whose origin-destination couple is "LHR, SYD".
- the date range must include the date "2011-06-10".
- the time range must include the time "21:45".
- the airline operating is "BA", so it must be the airline pricing.

We can deduce a part of our network rule file :

We have no information about stay duration and advance purchase (such information are contained into the booking request): so let us put "0" to embrace all the requests possible.

No information for the point-of-sale and the channel too: let us consider all the channels ("IN", "DN", "IF" and "DF") and all the points of sale (the origin "LHR", the destination "SYD" and the rest-of-the-world "ROW") existing. To access this information, we could look into the default booking request.

The input file is now:

Let us say we have just the Economy cabin "Y" and British Airways prices ticket for class "Y".

No information about the trip type, so we duplicate all the network rules for both type: one-way "OW" and round-trip "RT" (to access this information, we could look to the default booking request).

The network options are all set to a default value "T" (meaning true) and the network values are chosen to be all distinct.

We obtain:

15.4.2 Building the BOM tree with an input file

The steps are the same as before [Summary of the different steps](#) except the bom tree must be built using the network input file :

15.4.3 Result of the Batch Program

When the `AirSched.cpp` program is run with the `-f` option linking with the file built just above:

```
~/AirSched -f ~/<YourFileName>.csv
```

the last lines of the log output should look like:

```
[D]~/AirSchedgit/AirSched/batches/AirSched.cpp:223: Travel solutions:
    [0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- Y, 145, 1 1 1 ---
```

We have just one network option added to the travel solution. We can deduce from the price value 145 that the network builder used the network rule number 15 to price the travel solution. We have an inbound or outbound trip of a round trip: the total price 290 has been divided by 2.

16 Command-Line Test to Demonstrate How To Test the Airlnv Project

```
*/
// //////////////////////////////////////
// Import section
// //////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE InventoryTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/stdair_exceptions.hpp>
// Airlnv
#include <airlnv/AIRINV_Types.hpp>
#include <airlnv/AIRINV_Master_Service.hpp>
#include <airlnv/config/airlnv-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("InventoryTestSuite_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level
        (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

// //////////////////////////////////////
bool testInventoryHelper (const unsigned short iTestFlag,
                        const stdair::Filename_T& iInventoryInputFilename,
                        const stdair::Filename_T& iScheduleInputFilename,
```

```

        const stdair::Filename_T& iODInputFilename,
        const stdair::Filename_T& iYieldInputFilename,
        const bool isBuiltin,
        const bool isForSchedule) {

// Output log File
std::ostringstream oStr;
oStr << "InventoryTestSuite_" << iTestFlag << ".log";
const stdair::Filename_T lLogFilename (oStr.str());

// Set the log parameters
std::ofstream logOutputFile;
// Open and clean the log outputfile
logOutputFile.open (lLogFilename.c_str());
logOutputFile.clear();

// Initialise the AirInv service object
const bool lForceMultipleInit = true;
stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
                                logOutputFile,
                                lForceMultipleInit);

// Initialise the inventory service
AIRINV::AIRINV_Master_Service airinvService (lLogParams);

// Parameters for the sale
std::string lSegmentDateKey;
stdair::ClassCode_T lClassCode;
const stdair::PartySize_T lPartySize (2);

// Check whether or not a (CSV) input file should be read
if (isBuiltin == true) {

    // Build the default sample BOM tree (filled with inventories) for AirInv
    airinvService.buildSampleBom();

    // Define a specific segment-date key for the sample BOM tree
    lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
    lClassCode = "Q";

} else {

    if (isForSchedule == true) {
        // Build the BOM tree from parsing a schedule file (and O&D list)
        AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
        airinvService.parseAndLoad (iScheduleInputFilename, iODInputFilename,
                                    lYieldFilePath);

        // Define a specific segment-date key for the schedule-based inventory
        lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
        lClassCode = "Y";

    } else {

        // Build the BOM tree from parsing an inventory dump file
        airinvService.parseAndLoad (iInventoryInputFilename);

        // Define a specific segment-date key for the inventory parsed file
        //const std::string lSegmentDateKey ("SV, 5, 2010-03-11, KBP, JFK,
        //08:00:00");
        lSegmentDateKey = "SV, 5, 2010-03-11, KBP, JFK, 08:00:00";
        lClassCode = "J";
    }

}

// Make a booking
const bool hasSaleBeenSuccessful =
    airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);

// DEBUG: Display the list of travel solutions
const std::string& lCSVDump = airinvService.csvDisplay();
STDAIR_LOG_DEBUG (lCSVDump);

// Close the log file
logOutputFile.close();

if (hasSaleBeenSuccessful == false) {
    STDAIR_LOG_DEBUG ("No sale can be made for '" << lSegmentDateKey
                     << "'");
}

return hasSaleBeenSuccessful;
}

// ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

```

```
// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestFixture);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell) {

    // Input file name
    const stdair::Filename_T lInventoryInputFilename (STDAIR_SAMPLE_DIR
                                                       "/invdump01.csv");

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;
    // State whether the BOM tree should be built from a schedule file (instead
    // of from an inventory dump)
    const bool isForSchedule = false;

    // Try sell a default segment.
    bool hasTestBeenSuccessful = false;
    BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                           testInventoryHelper (0, lInventoryInputFilename,
                                                " ", " ", " ", isBuiltin,
                                                isForSchedule));
    BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
}

BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = true;
    // State whether the BOM tree should be built from a schedule file (instead
    // of from an inventory dump)
    const bool isForSchedule = false;

    // Try sell a default segment.
    bool hasTestBeenSuccessful = false;
    BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                           testInventoryHelper (1, " ", " ", " ", " ", " ",
                                                isBuiltin, isForSchedule));
    BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
}

BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_schedule) {

    // Input file names
    const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                       "/schedule01.csv");
    const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                                 "/ond01.csv");
    const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                                  "/yieldstore01.csv");

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;
    // State whether the BOM tree should be built from a schedule file (instead
    // of from an inventory dump)
    const bool isForSchedule = true;

    // Try sell a default segment.
    bool hasTestBeenSuccessful = false;
    BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                           testInventoryHelper (2, " ",
                                                lScheduleInputFilename,
                                                lODInputFilename,
                                                lYieldInputFilename,
                                                isBuiltin, isForSchedule));
    BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
}

BOOST_AUTO_TEST_CASE (airinv_error_inventory_input_file) {

    // Inventory input file name
    const stdair::Filename_T lMissingInventoryFilename (STDAIR_SAMPLE_DIR
                                                         "/missingFile.csv");

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;
    // State whether the BOM tree should be built from a schedule file (instead
    // of from an inventory dump)
    const bool isForSchedule = false;

```

```

// Try sell a default segment.
BOOST_CHECK_THROW (testInventoryHelper (3, lMissingInventoryFilename,
                                     " ", " ", " ", isBuiltin,
                                     isForSchedule),
                  AIRINV::InventoryInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airinv_error_schedule_input_file) {

// Schedule input file name
const stdair::Filename_T lMissingScheduleFilename (STDAIR_SAMPLE_DIR
                                                    "/missingFile.csv");

// State whether the BOM tree should be built-in or parsed from an input file
const bool isBuiltin = false;
// State whether the BOM tree should be built from a schedule file (instead
// of from an inventory dump)
const bool isForSchedule = true;

// Try sell a default segment.
BOOST_CHECK_THROW (testInventoryHelper (4, " ", lMissingScheduleFilename,
                                     " ", " ", isBuiltin, isForSchedule),
                  AIRINV::ScheduleInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airinv_error_yield_input_file) {

// Input file names
const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                  "/schedule01.csv");
const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                           "/ond01.csv");
const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                              "/missingFile.csv");

// State whether the BOM tree should be built-in or parsed from an input file
const bool isBuiltin = false;
// State whether the BOM tree should be built from a schedule file (instead
// of from an inventory dump)
const bool isForSchedule = true;

// Try sell a default segment.
BOOST_CHECK_THROW (testInventoryHelper (5, " ",
                                     lScheduleInputFilename,
                                     lODInputFilename,
                                     lYieldInputFilename,
                                     isBuiltin, isForSchedule),
                  AIRRAC::YieldInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airinv_error_flight_date_duplication) {

// Input file names
const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                  "/scheduleError01.csv");
const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                           "/ond01.csv");
const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                              "/missingFile.csv");

// State whether the BOM tree should be built-in or parsed from an input file
const bool isBuiltin = false;
// State whether the BOM tree should be built from a schedule file (instead
// of from an inventory dump)
const bool isForSchedule = true;

// Try sell a default segment.
BOOST_CHECK_THROW (testInventoryHelper (6, " ",
                                     lScheduleInputFilename,
                                     lODInputFilename,
                                     lYieldInputFilename,
                                     isBuiltin, isForSchedule),
                  AIRINV::FlightDateDuplicationException);
}

BOOST_AUTO_TEST_CASE (airinv_error_schedule_parsing_failed) {

// Input file names
const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                  "/scheduleError02.csv");
const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                           "/ond01.csv");
const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                              "/ond01.csv");

```

```

        "/yieldstore01.csv");

// State whether the BOM tree should be built-in or parsed from an input file
const bool isBuiltin = false;
// State whether the BOM tree should be built from a schedule file (instead
// of from an inventory dump)
const bool isForSchedule = true;

// Try sell a default segment.
BOOST_CHECK_THROW (testInventoryHelper (7, " ",
                                       lScheduleInputFilename,
                                       lODInputFilename,
                                       lYieldInputFilename,
                                       isBuiltin, isForSchedule),
                  AIRINV::ScheduleFileParsingFailedException);
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END ()

/*!

```

17 Directory Hierarchy

17.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

airinv	99
basic	99
batches	99
bom	99
command	100
vault	102
config	101
factory	101
server	101
service	102
ui	102
cmdline	100
test	102
airinv	98

18 Namespace Index

18.1 Namespace List

Here is a list of all namespaces with brief descriptions:

AIRINV	103
---------------	------------

AIRINV::DCPParserHelper	109
AIRINV::InventoryParserHelper	111
AIRINV::ScheduleParserHelper	114
stdair	
Forward declarations	117
swift	
The wrapper namespace	117

19 Class Index

19.1 Class Hierarchy

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

AIRINV::AIRINV_Master_Service	118
AIRINV::AIRINV_Service	123
std::basic_fstream< char >	
std::basic_fstream< wchar_t >	
std::basic_ifstream< char >	
std::basic_ifstream< wchar_t >	
std::basic_ios< char >	
std::basic_ios< wchar_t >	
std::basic_iostream< char >	
std::basic_iostream< wchar_t >	
std::basic_istream< char >	
std::basic_istream< wchar_t >	
std::basic_istreamstream< char >	
std::basic_istreamstream< wchar_t >	
std::basic_ofstream< char >	
std::basic_ofstream< wchar_t >	
std::basic_ostream< char >	
std::basic_ostream< wchar_t >	
std::basic_ostreamstream< char >	
std::basic_ostreamstream< wchar_t >	
std::basic_string< char >	
std::basic_string< wchar_t >	
std::basic_stringstream< char >	
std::basic_stringstream< wchar_t >	
AIRINV::BomAbstract	130
stdair::BomPropertyTree	132
AIRINV::BomRootHelper	133
AIRINV::BookingClassHelper	134
CmdAbstract	139
AIRINV::DCPEventGenerator	142
AIRINV::DCPParser	149
AIRINV::DCPRuleFileParser	150

AIRINV::FlightPeriodFileParser	188
AIRINV::InventoryBuilder	203
AIRINV::InventoryFileParser	204
AIRINV::InventoryGenerator	205
AIRINV::InventoryParser	210
AIRINV::ScheduleParser	231
COMMAND	140
AIRINV::DefaultMap	156
AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >	157
AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >	161
enable_shared_from_this	170
AIRINV::Connection	141
AIRINV::FacBomAbstract	173
FacServiceAbstract	175
AIRINV::FacAirinvMasterServiceContext	170
AIRINV::FacAirinvServiceContext	172
AIRINV::FacServiceAbstract	175
AIRINV::FacSupervisor	177
FileNotFoundException	181
AIRINV::InventoryInputFileNotFoundException	208
AIRINV::ScheduleInputFileNotFoundException	230
AIRINV::FlightDateHelper	182
grammar	202
AIRINV::InventoryParserHelper::InventoryParser	211
AIRINV::ScheduleParserHelper::FlightPeriodParser	189
grammar	202
AIRINV::DCPParserHelper::DCPRuleParser	151
AIRINV::GuillotineBlockHelper	202
AIRINV::header	203
AIRINV::InventoryHelper	206
AIRINV::InventoryManager	208
AIRINV::LegCabinHelper	213

noncopyable	218
AIRINV::AirInvServer	129
AIRINV::Connection	141
AIRINV::RequestHandler	227
ObjectCreationgDuplicationException	219
AIRINV::FlightDateDuplicationException	181
ParserException	219
AIRINV::SegmentDateNotFoundException	236
AIRINV::InventoryParserHelper::ParserSemanticAction	219
AIRINV::InventoryParserHelper::doEndFlightDate	168
AIRINV::InventoryParserHelper::storeACP	246
AIRINV::InventoryParserHelper::storeAirlineCode	249
AIRINV::InventoryParserHelper::storeAU	253
AIRINV::InventoryParserHelper::storeBoardingDate	255
AIRINV::InventoryParserHelper::storeBoardingTime	256
AIRINV::InventoryParserHelper::storeBookingCounter	259
AIRINV::InventoryParserHelper::storeBucketAvaibility	261
AIRINV::InventoryParserHelper::storeClassAvailability	269
AIRINV::InventoryParserHelper::storeClassCode	270
AIRINV::InventoryParserHelper::storeClassETB	273
AIRINV::InventoryParserHelper::storeCumulatedProtection	275
AIRINV::InventoryParserHelper::storeETB	289
AIRINV::InventoryParserHelper::storeFamilyCode	291
AIRINV::InventoryParserHelper::storeFClasses	294
AIRINV::InventoryParserHelper::storeFlightDate	296
AIRINV::InventoryParserHelper::storeFlightNumber	298
AIRINV::InventoryParserHelper::storeFlightTypeCode	301
AIRINV::InventoryParserHelper::storeFlightVisibilityCode	302
AIRINV::InventoryParserHelper::storeGAV	304
AIRINV::InventoryParserHelper::storeLegBoardingPoint	306
AIRINV::InventoryParserHelper::storeLegCabinCode	310
AIRINV::InventoryParserHelper::storeLegOffPoint	313

AIRINV::InventoryParserHelper::storeNAV	316
AIRINV::InventoryParserHelper::storeNbOfBkgs	317
AIRINV::InventoryParserHelper::storeNbOfGroupBkgs	319
AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs	320
AIRINV::InventoryParserHelper::storeNbOfStaffBkgs	322
AIRINV::InventoryParserHelper::storeNbOfWLBkgs	324
AIRINV::InventoryParserHelper::storeNego	325
AIRINV::InventoryParserHelper::storeNoShow	328
AIRINV::InventoryParserHelper::storeOffDate	329
AIRINV::InventoryParserHelper::storeOffTime	331
AIRINV::InventoryParserHelper::storeOverbooking	335
AIRINV::InventoryParserHelper::storeParentClassCode	337
AIRINV::InventoryParserHelper::storeParentSubclassCode	338
AIRINV::InventoryParserHelper::storeProtection	341
AIRINV::InventoryParserHelper::storeRevenueAvailability	343
AIRINV::InventoryParserHelper::storeSaleableCapacity	344
AIRINV::InventoryParserHelper::storeSeatIndex	347
AIRINV::InventoryParserHelper::storeSegmentAvailability	349
AIRINV::InventoryParserHelper::storeSegmentBoardingPoint	350
AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter	353
AIRINV::InventoryParserHelper::storeSegmentCabinCode	355
AIRINV::InventoryParserHelper::storeSegmentOffPoint	358
AIRINV::InventoryParserHelper::storeSnapshotDate	362
AIRINV::InventoryParserHelper::storeSubclassCode	365
AIRINV::InventoryParserHelper::storeUPR	366
AIRINV::InventoryParserHelper::storeYieldUpperRange	368
AIRINV::ScheduleParserHelper::ParserSemanticAction	221
AIRINV::ScheduleParserHelper::doEndFlight	166
AIRINV::ScheduleParserHelper::storeAirlineCode	251
AIRINV::ScheduleParserHelper::storeBoardingTime	258
AIRINV::ScheduleParserHelper::storeCapacity	264
AIRINV::ScheduleParserHelper::storeClasses	272

AIRINV::ScheduleParserHelper::storeDateRangeEnd	278
AIRINV::ScheduleParserHelper::storeDateRangeStart	279
AIRINV::ScheduleParserHelper::storeDow	285
AIRINV::ScheduleParserHelper::storeElapsedTime	286
AIRINV::ScheduleParserHelper::storeFamilyCode	292
AIRINV::ScheduleParserHelper::storeFClasses	295
AIRINV::ScheduleParserHelper::storeFlightNumber	300
AIRINV::ScheduleParserHelper::storeLegBoardingPoint	307
AIRINV::ScheduleParserHelper::storeLegCabinCode	309
AIRINV::ScheduleParserHelper::storeLegOffPoint	311
AIRINV::ScheduleParserHelper::storeOffTime	333
AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint	352
AIRINV::ScheduleParserHelper::storeSegmentCabinCode	357
AIRINV::ScheduleParserHelper::storeSegmentOffPoint	359
AIRINV::ScheduleParserHelper::storeSegmentSpecificity	361
AIRINV::DCPParserHelper::ParserSemanticAction	223
AIRINV::DCPParserHelper::doEndDCP	165
AIRINV::DCPParserHelper::storeAdvancePurchase	248
AIRINV::DCPParserHelper::storeAirlineCode	252
AIRINV::DCPParserHelper::storeCabinCode	262
AIRINV::DCPParserHelper::storeChangeFees	265
AIRINV::DCPParserHelper::storeChannel	266
AIRINV::DCPParserHelper::storeClass	267
AIRINV::DCPParserHelper::storeDateRangeEnd	276
AIRINV::DCPParserHelper::storeDateRangeStart	280
AIRINV::DCPParserHelper::storeDCP	282
AIRINV::DCPParserHelper::storeDCPId	283
AIRINV::DCPParserHelper::storeDestination	284
AIRINV::DCPParserHelper::storeEndRangeTime	288
AIRINV::DCPParserHelper::storeMinimumStay	314
AIRINV::DCPParserHelper::storeNonRefundable	327
AIRINV::DCPParserHelper::storeOrigin	334

AIRINV::DCPParserHelper::storePOS	340
AIRINV::DCPParserHelper::storeSaturdayStay	346
AIRINV::DCPParserHelper::storeStartRangeTime	364
ParsingFileFailedException	225
AIRINV::InventoryFileParsingFailedException	205
AIRINV::ScheduleFileParsingFailedException	230
AIRINV::Reply	225
AIRINV::Request	226
AIRINV::RequestParser	228
RootException	229
AIRINV::BookingException	137
AIRINV::SegmentCabinHelper	232
AIRINV::SegmentDateHelper	235
ServiceAbstract	238
AIRINV::AIRINV_Master_ServiceContext	123
AIRINV::AIRINV_ServiceContext	129
AIRINV::ServiceAbstract	238
swift::SKeymap	240
swift::SReadline	242
StructAbstract	370
AIRINV::BookingClassStruct	134
AIRINV::BucketStruct	138
AIRINV::DCPEventStruct	143
AIRINV::FareFamilyStruct	179
AIRINV::FlightDateStruct	183
AIRINV::FlightPeriodStruct	191
AIRINV::FlightRequestStatus	196
AIRINV::FlightTypeCode	198
AIRINV::FlightVisibilityCode	200
AIRINV::LegCabinStruct	213
AIRINV::LegStruct	216
AIRINV::SegmentCabinStruct	233

AIRINV::SegmentStruct	236
TestFixture	370
InventoryTestSuite	212

20 Class Index

20.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AIRINV::AIRINV_Master_Service Interface for the AIRINV Services	118
AIRINV::AIRINV_Master_ServiceContext	123
AIRINV::AIRINV_Service Interface for the AIRINV Services	123
AIRINV::AIRINV_ServiceContext Class holding the context of the AirInv services	129
AIRINV::AirInvServer	129
AIRINV::BomAbstract	130
stdair::BomPropertyTree	132
AIRINV::BomRootHelper	133
AIRINV::BookingClassHelper	134
AIRINV::BookingClassStruct	134
AIRINV::BookingException	137
AIRINV::BucketStruct Utility Structure for the parsing of Bucket structures	138
CmdAbstract	139
COMMAND	140
AIRINV::Connection	141
AIRINV::DCPEventGenerator	142
AIRINV::DCPEventStruct	143
AIRINV::DCPParser	149
AIRINV::DCPRuleFileParser	150
AIRINV::DCPParserHelper::DCPRuleParser	151
AIRINV::DefaultMap	156
AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >	157

AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >	161
AIRINV::DCPParserHelper::doEndDCP	165
AIRINV::ScheduleParserHelper::doEndFlight	166
AIRINV::InventoryParserHelper::doEndFlightDate	168
enable_shared_from_this	170
AIRINV::FacAirinvMasterServiceContext Factory for Bucket	170
AIRINV::FacAirinvServiceContext	172
AIRINV::FacBomAbstract	173
FacServiceAbstract	175
AIRINV::FacServiceAbstract	175
AIRINV::FacSupervisor	177
AIRINV::FareFamilyStruct Utility Structure for the parsing of fare family details	179
FileNotFoundException	181
AIRINV::FlightDateDuplicationException	181
AIRINV::FlightDateHelper	182
AIRINV::FlightDateStruct	183
AIRINV::FlightPeriodFileParser	188
AIRINV::ScheduleParserHelper::FlightPeriodParser	189
AIRINV::FlightPeriodStruct	191
AIRINV::FlightRequestStatus	196
AIRINV::FlightTypeCode	198
AIRINV::FlightVisibilityCode	200
grammar	202
grammar	202
AIRINV::GuillotineBlockHelper	202
AIRINV::header	203
AIRINV::InventoryBuilder Class handling the generation / instantiation of the Inventory BOM	203
AIRINV::InventoryFileParser	204
AIRINV::InventoryFileParsingFailedException	205
AIRINV::InventoryGenerator Class handling the generation / instantiation of the Inventory BOM	205

AIRINV::InventoryHelper	206
AIRINV::InventoryInputFileNotFoundException	208
AIRINV::InventoryManager	208
AIRINV::InventoryParser	
Class wrapping the parser entry point	210
AIRINV::InventoryParserHelper::InventoryParser	211
InventoryTestSuite	212
AIRINV::LegCabinHelper	213
AIRINV::LegCabinStruct	213
AIRINV::LegStruct	216
noncopyable	218
ObjectCreationgDuplicationException	219
ParserException	219
AIRINV::InventoryParserHelper::ParserSemanticAction	219
AIRINV::ScheduleParserHelper::ParserSemanticAction	221
AIRINV::DCPParserHelper::ParserSemanticAction	223
ParsingFileFailedException	225
AIRINV::Reply	225
AIRINV::Request	226
AIRINV::RequestHandler	
The common handler for all incoming requests	227
AIRINV::RequestParser	
Parser for incoming requests	228
RootException	229
AIRINV::ScheduleFileParsingFailedException	230
AIRINV::ScheduleInputFileNotFoundException	230
AIRINV::ScheduleParser	
Class wrapping the parser entry point	231
AIRINV::SegmentCabinHelper	
Class representing the actual business functions for an airline segment-cabin	232
AIRINV::SegmentCabinStruct	
Utility Structure for the parsing of SegmentCabin details	233
AIRINV::SegmentDateHelper	235
AIRINV::SegmentDateNotFoundException	236
AIRINV::SegmentStruct	236

ServiceAbstract	238
AIRINV::ServiceAbstract	238
swift::SKeymap The readline keymap wrapper	240
swift::SReadline The readline library wrapper	242
AIRINV::InventoryParserHelper::storeACP	246
AIRINV::DCPParserHelper::storeAdvancePurchase	248
AIRINV::InventoryParserHelper::storeAirlineCode	249
AIRINV::ScheduleParserHelper::storeAirlineCode	251
AIRINV::DCPParserHelper::storeAirlineCode	252
AIRINV::InventoryParserHelper::storeAU	253
AIRINV::InventoryParserHelper::storeBoardingDate	255
AIRINV::InventoryParserHelper::storeBoardingTime	256
AIRINV::ScheduleParserHelper::storeBoardingTime	258
AIRINV::InventoryParserHelper::storeBookingCounter	259
AIRINV::InventoryParserHelper::storeBucketAvaibility	261
AIRINV::DCPParserHelper::storeCabinCode	262
AIRINV::ScheduleParserHelper::storeCapacity	264
AIRINV::DCPParserHelper::storeChangeFees	265
AIRINV::DCPParserHelper::storeChannel	266
AIRINV::DCPParserHelper::storeClass	267
AIRINV::InventoryParserHelper::storeClassAvailability	269
AIRINV::InventoryParserHelper::storeClassCode	270
AIRINV::ScheduleParserHelper::storeClasses	272
AIRINV::InventoryParserHelper::storeClassETB	273
AIRINV::InventoryParserHelper::storeCumulatedProtection	275
AIRINV::DCPParserHelper::storeDateRangeEnd	276
AIRINV::ScheduleParserHelper::storeDateRangeEnd	278
AIRINV::ScheduleParserHelper::storeDateRangeStart	279
AIRINV::DCPParserHelper::storeDateRangeStart	280
AIRINV::DCPParserHelper::storeDCP	282
AIRINV::DCPParserHelper::storeDCPIId	283

AIRINV::DCPParserHelper::storeDestination	284
AIRINV::ScheduleParserHelper::storeDow	285
AIRINV::ScheduleParserHelper::storeElapsedTime	286
AIRINV::DCPParserHelper::storeEndRangeTime	288
AIRINV::InventoryParserHelper::storeETB	289
AIRINV::InventoryParserHelper::storeFamilyCode	291
AIRINV::ScheduleParserHelper::storeFamilyCode	292
AIRINV::InventoryParserHelper::storeFClasses	294
AIRINV::ScheduleParserHelper::storeFClasses	295
AIRINV::InventoryParserHelper::storeFlightDate	296
AIRINV::InventoryParserHelper::storeFlightNumber	298
AIRINV::ScheduleParserHelper::storeFlightNumber	300
AIRINV::InventoryParserHelper::storeFlightTypeCode	301
AIRINV::InventoryParserHelper::storeFlightVisibilityCode	302
AIRINV::InventoryParserHelper::storeGAV	304
AIRINV::InventoryParserHelper::storeLegBoardingPoint	306
AIRINV::ScheduleParserHelper::storeLegBoardingPoint	307
AIRINV::ScheduleParserHelper::storeLegCabinCode	309
AIRINV::InventoryParserHelper::storeLegCabinCode	310
AIRINV::ScheduleParserHelper::storeLegOffPoint	311
AIRINV::InventoryParserHelper::storeLegOffPoint	313
AIRINV::DCPParserHelper::storeMinimumStay	314
AIRINV::InventoryParserHelper::storeNAV	316
AIRINV::InventoryParserHelper::storeNbOfBkgs	317
AIRINV::InventoryParserHelper::storeNbOfGroupBkgs	319
AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs	320
AIRINV::InventoryParserHelper::storeNbOfStaffBkgs	322
AIRINV::InventoryParserHelper::storeNbOfWLBkgs	324
AIRINV::InventoryParserHelper::storeNego	325
AIRINV::DCPParserHelper::storeNonRefundable	327
AIRINV::InventoryParserHelper::storeNoShow	328
AIRINV::InventoryParserHelper::storeOffDate	329

AIRINV::InventoryParserHelper::storeOffTime	331
AIRINV::ScheduleParserHelper::storeOffTime	333
AIRINV::DCPParserHelper::storeOrigin	334
AIRINV::InventoryParserHelper::storeOverbooking	335
AIRINV::InventoryParserHelper::storeParentClassCode	337
AIRINV::InventoryParserHelper::storeParentSubclassCode	338
AIRINV::DCPParserHelper::storePOS	340
AIRINV::InventoryParserHelper::storeProtection	341
AIRINV::InventoryParserHelper::storeRevenueAvailability	343
AIRINV::InventoryParserHelper::storeSaleableCapacity	344
AIRINV::DCPParserHelper::storeSaturdayStay	346
AIRINV::InventoryParserHelper::storeSeatIndex	347
AIRINV::InventoryParserHelper::storeSegmentAvailability	349
AIRINV::InventoryParserHelper::storeSegmentBoardingPoint	350
AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint	352
AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter	353
AIRINV::InventoryParserHelper::storeSegmentCabinCode	355
AIRINV::ScheduleParserHelper::storeSegmentCabinCode	357
AIRINV::InventoryParserHelper::storeSegmentOffPoint	358
AIRINV::ScheduleParserHelper::storeSegmentOffPoint	359
AIRINV::ScheduleParserHelper::storeSegmentSpecificity	361
AIRINV::InventoryParserHelper::storeSnapshotDate	362
AIRINV::DCPParserHelper::storeStartRangeTime	364
AIRINV::InventoryParserHelper::storeSubclassCode	365
AIRINV::InventoryParserHelper::storeUPR	366
AIRINV::InventoryParserHelper::storeYieldUpperRange	368
StructAbstract	370
TestFixture	370

21 File Index

21.1 File List

Here is a list of all files with brief descriptions:

airinv/ AIRINV_Master_Service.hpp	371
airinv/ AIRINV_Service.hpp	373
airinv/ AIRINV_Types.hpp	375
airinv/ FlightRequestStatus.hpp	536
airinv/basic/ BasConst.cpp	376
airinv/basic/ BasConst_AIRINV_Service.hpp	378
airinv/basic/ BasConst_Curves.hpp	378
airinv/basic/ BasConst_General.hpp	379
airinv/basic/ BasParserTypes.hpp	380
airinv/basic/ FlightRequestStatus.cpp	381
airinv/basic/ FlightTypeCode.cpp	383
airinv/basic/ FlightTypeCode.hpp	384
airinv/basic/ FlightVisibilityCode.cpp	385
airinv/basic/ FlightVisibilityCode.hpp	387
airinv/batches/ airinv_parseInventory.cpp	387
airinv/batches/ parseInventory.cpp	391
airinv/bom/ AirportList.hpp	395
airinv/bom/ BomAbstract.cpp	396
airinv/bom/ BomAbstract.hpp	397
airinv/bom/ BomRootHelper.cpp	398
airinv/bom/ BomRootHelper.hpp	398
airinv/bom/ BookingClassHelper.cpp	399
airinv/bom/ BookingClassHelper.hpp	399
airinv/bom/ BookingClassStruct.cpp	400
airinv/bom/ BookingClassStruct.hpp	401
airinv/bom/ BucketStruct.cpp	402
airinv/bom/ BucketStruct.hpp	403
airinv/bom/ DCPEventStruct.cpp	404
airinv/bom/ DCPEventStruct.hpp	406
airinv/bom/ FareFamilyStruct.cpp	408
airinv/bom/ FareFamilyStruct.hpp	409
airinv/bom/ FlightDateHelper.cpp	410

airinv/bom/FlightDateHelper.hpp	411
airinv/bom/FlightDateStruct.cpp	412
airinv/bom/FlightDateStruct.hpp	416
airinv/bom/FlightPeriodStruct.cpp	417
airinv/bom/FlightPeriodStruct.hpp	421
airinv/bom/GuillotineBlockHelper.cpp	422
airinv/bom/GuillotineBlockHelper.hpp	425
airinv/bom/InventoryHelper.cpp	426
airinv/bom/InventoryHelper.hpp	431
airinv/bom/LegCabinHelper.cpp	432
airinv/bom/LegCabinHelper.hpp	432
airinv/bom/LegCabinStruct.cpp	433
airinv/bom/LegCabinStruct.hpp	434
airinv/bom/LegStruct.cpp	435
airinv/bom/LegStruct.hpp	436
airinv/bom/SegmentCabinHelper.cpp	437
airinv/bom/SegmentCabinHelper.hpp	440
airinv/bom/SegmentCabinStruct.cpp	441
airinv/bom/SegmentCabinStruct.hpp	442
airinv/bom/SegmentDateHelper.cpp	442
airinv/bom/SegmentDateHelper.hpp	444
airinv/bom/SegmentStruct.cpp	445
airinv/bom/SegmentStruct.hpp	446
airinv/command/InventoryBuilder.cpp	447
airinv/command/InventoryBuilder.hpp	451
airinv/command/InventoryGenerator.cpp	453
airinv/command/InventoryGenerator.hpp	457
airinv/command/InventoryManager.cpp	459
airinv/command/InventoryManager.hpp	473
airinv/command/InventoryParser.cpp	474
airinv/command/InventoryParser.hpp	475
airinv/command/InventoryParserHelper.cpp	476

airinv/command/InventoryParserHelper.hpp	491
airinv/command/ScheduleParser.cpp	495
airinv/command/ScheduleParser.hpp	496
airinv/command/ScheduleParserHelper.cpp	497
airinv/command/ScheduleParserHelper.hpp	506
airinv/command/vault/DCPEventGenerator.cpp	509
airinv/command/vault/DCPEventGenerator.hpp	510
airinv/command/vault/DCPParser.cpp	511
airinv/command/vault/DCPParser.hpp	511
airinv/command/vault/DCPParserHelper.cpp	512
airinv/command/vault/DCPParserHelper.hpp	521
airinv/config/airinv-paths.hpp	525
airinv/config/airinv-paths.hpp.in	527
airinv/factory/FacAirinvMasterServiceContext.cpp	528
airinv/factory/FacAirinvMasterServiceContext.hpp	528
airinv/factory/FacAirinvServiceContext.cpp	529
airinv/factory/FacAirinvServiceContext.hpp	530
airinv/factory/FacBomAbstract.cpp	531
airinv/factory/FacBomAbstract.hpp	532
airinv/factory/FacServiceAbstract.cpp	533
airinv/factory/FacServiceAbstract.hpp	533
airinv/factory/FacSupervisor.cpp	534
airinv/factory/FacSupervisor.hpp	535
airinv/server/AirInvClient.cpp	537
airinv/server/AirInvClient_ASIO.cpp	538
airinv/server/AirInvServer.cpp	539
airinv/server/AirInvServer.hpp	544
airinv/server/AirInvServer_ASIO.cpp	545
airinv/server/BomPropertyTree.cpp	547
airinv/server/BomPropertyTree.hpp	548
airinv/server/Connection.cpp	549
airinv/server/Connection.hpp	551

airinv/server/header.hpp	552
airinv/server/posix_main.cpp	552
airinv/server/Reply.cpp	553
airinv/server/Reply.hpp	554
airinv/server/Request.cpp	555
airinv/server/Request.hpp	555
airinv/server/RequestHandler.cpp	556
airinv/server/RequestHandler.hpp	557
airinv/server/RequestParser.cpp	558
airinv/server/RequestParser.hpp	561
airinv/server/win_main.cpp	562
airinv/service/AIRINV_Master_Service.cpp	564
airinv/service/AIRINV_Master_ServiceContext.cpp	571
airinv/service/AIRINV_Master_ServiceContext.hpp	572
airinv/service/AIRINV_Service.cpp	574
airinv/service/AIRINV_ServiceContext.cpp	582
airinv/service/AIRINV_ServiceContext.hpp	583
airinv/service/ServiceAbstract.cpp	584
airinv/service/ServiceAbstract.hpp	585
airinv/ui/cmdline/airinv.cpp	586
airinv/ui/cmdline/readline_autocomp.hpp	602
airinv/ui/cmdline/SReadline.hpp C++ wrapper around libreadline	606
test/airinv/InventoryTestSuite.cpp	612
test/airinv/InventoryTestSuite.hpp	617

22 Directory Documentation

22.1 test/airinv/ Directory Reference

Files

- file [InventoryTestSuite.cpp](#)
- file [InventoryTestSuite.hpp](#)

22.2 airinv/ Directory Reference

Directories

- directory [basic](#)
- directory [batches](#)
- directory [bom](#)
- directory [command](#)
- directory [config](#)
- directory [factory](#)
- directory [server](#)
- directory [service](#)
- directory [ui](#)

Files

- file [AIRINV_Master_Service.hpp](#)
- file [AIRINV_Service.hpp](#)
- file [AIRINV_Types.hpp](#)
- file [FlightRequestStatus.hpp](#)

22.3 airinv/basic/ Directory Reference

Files

- file [BasConst.cpp](#)
- file [BasConst_AIRINV_Service.hpp](#)
- file [BasConst_Curves.hpp](#)
- file [BasConst_General.hpp](#)
- file [BasParserTypes.hpp](#)
- file [FlightRequestStatus.cpp](#)
- file [FlightTypeCode.cpp](#)
- file [FlightTypeCode.hpp](#)
- file [FlightVisibilityCode.cpp](#)
- file [FlightVisibilityCode.hpp](#)

22.4 airinv/batches/ Directory Reference

Files

- file [airinv_parseInventory.cpp](#)
- file [parseInventory.cpp](#)

22.5 airinv/bom/ Directory Reference

Files

- file [AirportList.hpp](#)
- file [BomAbstract.cpp](#)
- file [BomAbstract.hpp](#)
- file [BomRootHelper.cpp](#)
- file [BomRootHelper.hpp](#)
- file [BookingClassHelper.cpp](#)

- file [BookingClassHelper.hpp](#)
- file [BookingClassStruct.cpp](#)
- file [BookingClassStruct.hpp](#)
- file [BucketStruct.cpp](#)
- file [BucketStruct.hpp](#)
- file [DCPEventStruct.cpp](#)
- file [DCPEventStruct.hpp](#)
- file [FareFamilyStruct.cpp](#)
- file [FareFamilyStruct.hpp](#)
- file [FlightDateHelper.cpp](#)
- file [FlightDateHelper.hpp](#)
- file [FlightDateStruct.cpp](#)
- file [FlightDateStruct.hpp](#)
- file [FlightPeriodStruct.cpp](#)
- file [FlightPeriodStruct.hpp](#)
- file [GuillotineBlockHelper.cpp](#)
- file [GuillotineBlockHelper.hpp](#)
- file [InventoryHelper.cpp](#)
- file [InventoryHelper.hpp](#)
- file [LegCabinHelper.cpp](#)
- file [LegCabinHelper.hpp](#)
- file [LegCabinStruct.cpp](#)
- file [LegCabinStruct.hpp](#)
- file [LegStruct.cpp](#)
- file [LegStruct.hpp](#)
- file [SegmentCabinHelper.cpp](#)
- file [SegmentCabinHelper.hpp](#)
- file [SegmentCabinStruct.cpp](#)
- file [SegmentCabinStruct.hpp](#)
- file [SegmentDateHelper.cpp](#)
- file [SegmentDateHelper.hpp](#)
- file [SegmentStruct.cpp](#)
- file [SegmentStruct.hpp](#)

22.6 `airinv/ui/cmdline/` Directory Reference

Files

- file [airinv.cpp](#)
- file [readline_autocomp.hpp](#)
- file [SReadline.hpp](#)
C++ wrapper around libreadline.

22.7 `airinv/command/` Directory Reference

Directories

- directory [vault](#)

Files

- file [InventoryBuilder.cpp](#)
- file [InventoryBuilder.hpp](#)
- file [InventoryGenerator.cpp](#)
- file [InventoryGenerator.hpp](#)
- file [InventoryManager.cpp](#)
- file [InventoryManager.hpp](#)
- file [InventoryParser.cpp](#)
- file [InventoryParser.hpp](#)
- file [InventoryParserHelper.cpp](#)
- file [InventoryParserHelper.hpp](#)
- file [ScheduleParser.cpp](#)
- file [ScheduleParser.hpp](#)
- file [ScheduleParserHelper.cpp](#)
- file [ScheduleParserHelper.hpp](#)

22.8 airinv/config/ Directory Reference

Files

- file [airinv-paths.hpp](#)
- file [airinv-paths.hpp.in](#)

22.9 airinv/factory/ Directory Reference

Files

- file [FacAirinvMasterServiceContext.cpp](#)
- file [FacAirinvMasterServiceContext.hpp](#)
- file [FacAirinvServiceContext.cpp](#)
- file [FacAirinvServiceContext.hpp](#)
- file [FacBomAbstract.cpp](#)
- file [FacBomAbstract.hpp](#)
- file [FacServiceAbstract.cpp](#)
- file [FacServiceAbstract.hpp](#)
- file [FacSupervisor.cpp](#)
- file [FacSupervisor.hpp](#)

22.10 airinv/server/ Directory Reference

Files

- file [AirInvClient.cpp](#)
- file [AirInvClient_ASIO.cpp](#)
- file [AirInvServer.cpp](#)
- file [AirInvServer.hpp](#)
- file [AirInvServer_ASIO.cpp](#)
- file [BomPropertyTree.cpp](#)
- file [BomPropertyTree.hpp](#)
- file [Connection.cpp](#)
- file [Connection.hpp](#)
- file [header.hpp](#)

- file [posix_main.cpp](#)
- file [Reply.cpp](#)
- file [Reply.hpp](#)
- file [Request.cpp](#)
- file [Request.hpp](#)
- file [RequestHandler.cpp](#)
- file [RequestHandler.hpp](#)
- file [RequestParser.cpp](#)
- file [RequestParser.hpp](#)
- file [win_main.cpp](#)

22.11 airinv/service/ Directory Reference

Files

- file [AIRINV_Master_Service.cpp](#)
- file [AIRINV_Master_ServiceContext.cpp](#)
- file [AIRINV_Master_ServiceContext.hpp](#)
- file [AIRINV_Service.cpp](#)
- file [AIRINV_ServiceContext.cpp](#)
- file [AIRINV_ServiceContext.hpp](#)
- file [ServiceAbstract.cpp](#)
- file [ServiceAbstract.hpp](#)

22.12 test/ Directory Reference

Directories

- directory [airinv](#)

22.13 airinv/ui/ Directory Reference

Directories

- directory [cmdline](#)

22.14 airinv/command/vault/ Directory Reference

Files

- file [DCPEventGenerator.cpp](#)
- file [DCPEventGenerator.hpp](#)
- file [DCPParser.cpp](#)
- file [DCPParser.hpp](#)
- file [DCPParserHelper.cpp](#)
- file [DCPParserHelper.hpp](#)

23 Namespace Documentation

23.1 AIRINV Namespace Reference

Namespaces

- namespace [InventoryParserHelper](#)
- namespace [ScheduleParserHelper](#)
- namespace [DCPParserHelper](#)

Classes

- class [AIRINV_Master_Service](#)
Interface for the [AIRINV](#) Services.
- class [AIRINV_Service](#)
Interface for the [AIRINV](#) Services.
- class [InventoryFileParsingFailedException](#)
- class [ScheduleFileParsingFailedException](#)
- class [SegmentDateNotFoundException](#)
- class [InventoryInputFileNotFoundException](#)
- class [ScheduleInputFileNotFoundException](#)
- class [FlightDateDuplicationException](#)
- class [BookingException](#)
- struct [DefaultMap](#)
- struct [FlightTypeCode](#)
- struct [FlightVisibilityCode](#)
- class [BomAbstract](#)
- class [BomRootHelper](#)
- class [BookingClassHelper](#)
- struct [BookingClassStruct](#)
- struct [BucketStruct](#)
Utility Structure for the parsing of Bucket structures.
- struct [DCPEventStruct](#)
- struct [FareFamilyStruct](#)
Utility Structure for the parsing of fare family details.
- class [FlightDateHelper](#)
- struct [FlightDateStruct](#)
- struct [FlightPeriodStruct](#)
- class [GuillotineBlockHelper](#)
- class [InventoryHelper](#)
- class [LegCabinHelper](#)
- struct [LegCabinStruct](#)
- struct [LegStruct](#)
- class [SegmentCabinHelper](#)
Class representing the actual business functions for an airline segment-cabin.
- struct [SegmentCabinStruct](#)
Utility Structure for the parsing of SegmentCabin details.
- class [SegmentDateHelper](#)
- struct [SegmentStruct](#)
- class [InventoryBuilder](#)
Class handling the generation / instantiation of the Inventory BOM.
- class [InventoryGenerator](#)
Class handling the generation / instantiation of the Inventory BOM.

- class [InventoryManager](#)
- class [InventoryParser](#)
 - Class wrapping the parser entry point.*
- class [InventoryFileParser](#)
- class [ScheduleParser](#)
 - Class wrapping the parser entry point.*
- class [FlightPeriodFileParser](#)
- class [DCPEventGenerator](#)
- class [DCPParser](#)
- class [DCPRuleFileParser](#)
- class [FacAirinvMasterServiceContext](#)
 - Factory for Bucket.*
- class [FacAirinvServiceContext](#)
- class [FacBomAbstract](#)
- class [FacServiceAbstract](#)
- class [FacSupervisor](#)
- struct [FlightRequestStatus](#)
- class [AirInvServer](#)
- class [Connection](#)
- struct [header](#)
- struct [Reply](#)
- struct [Request](#)
- class [RequestHandler](#)
 - The common handler for all incoming requests.*
- class [RequestParser](#)
 - Parser for incoming requests.*
- class [AIRINV_Master_ServiceContext](#)
- class [AIRINV_ServiceContext](#)
 - Class holding the context of the AirInv services.*
- class [ServiceAbstract](#)

Typedefs

- typedef boost::shared_ptr
 < [AIRINV_Service](#) > [AIRINV_ServicePtr_T](#)
- typedef boost::shared_ptr
 < [AIRINV_Master_Service](#) > [AIRINV_Master_ServicePtr_T](#)
- typedef std::map< const
 stdair::AirlineCode_T,
 [AIRINV_ServicePtr_T](#) > [AIRINV_ServicePtr_Map_T](#)
- typedef std::map< const
 stdair::DTD_T, double > [FRAT5Curve_T](#)
- typedef char [char_t](#)
- typedef
 boost::spirit::classic::file_iterator
 < [char_t](#) > [iterator_t](#)
- typedef
 boost::spirit::classic::scanner
 < [iterator_t](#) > [scanner_t](#)
- typedef
 boost::spirit::classic::rule
 < [scanner_t](#) > [rule_t](#)
- typedef
 boost::spirit::classic::int_parser
 < unsigned int, 10, 1, 1 > [int1_p_t](#)

- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 2, 2 > [uint2_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 2 > [uint1_2_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 3 > [uint1_3_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 4, 4 > [uint4_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 4 > [uint1_4_p_t](#)
- typedef
boost::spirit::classic::chset
< [char_t](#) > [chset_t](#)
- typedef
boost::spirit::classic::impl::loop_traits
< [chset_t](#), unsigned int,
unsigned int >::type [repeat_p_t](#)
- typedef
boost::spirit::classic::bounded
< [uint2_p_t](#), unsigned int > [bounded2_p_t](#)
- typedef
boost::spirit::classic::bounded
< [uint1_2_p_t](#), unsigned int > [bounded1_2_p_t](#)
- typedef
boost::spirit::classic::bounded
< [uint1_3_p_t](#), unsigned int > [bounded1_3_p_t](#)
- typedef
boost::spirit::classic::bounded
< [uint4_p_t](#), unsigned int > [bounded4_p_t](#)
- typedef
boost::spirit::classic::bounded
< [uint1_4_p_t](#), unsigned int > [bounded1_4_p_t](#)
- typedef std::set
< stdair::AirportCode_T > [AirportList_T](#)
- typedef std::vector
< stdair::AirportCode_T > [AirportOrderedList_T](#)
- typedef std::vector
< [BookingClassStruct](#) > [BookingClassStructList_T](#)
- typedef std::vector< [BucketStruct](#) > [BucketStructList_T](#)
- typedef std::vector
< [FareFamilyStruct](#) > [FareFamilyStructList_T](#)
- typedef std::vector
< [LegCabinStruct](#) > [LegCabinStructList_T](#)
- typedef std::vector< [LegStruct](#) > [LegStructList_T](#)
- typedef std::vector
< [SegmentCabinStruct](#) > [SegmentCabinStructList_T](#)
- typedef std::vector
< [SegmentStruct](#) > [SegmentStructList_T](#)
- typedef std::map< const
stdair::Date_T,
stdair::SegmentCabin * > [DepartureDateSegmentCabinMap_T](#)

- typedef std::map< const std::string, [DepartureDateSegmentCabinMap_T](#) > [SimilarSegmentCabinSetMap_T](#)
- typedef boost::shared_ptr< boost::thread > [ThreadShrPtr_T](#)
- typedef std::vector< [ThreadShrPtr_T](#) > [ThreadShrPtrList_T](#)
- typedef boost::shared_ptr< [Connection](#) > [ConnectionShrPtr_T](#)

Variables

- const std::string [DEFAULT_AIRLINE_CODE](#) = "BA"
- const [FRAT5Curve_T](#) [DEFAULT_PICKUP_FRAT5_CURVE](#)

23.1.1 Typedef Documentation

23.1.1.1 typedef boost::shared_ptr<[AIRINV_Service](#)> [AIRINV::AIRINV_ServicePtr_T](#)

(Smart) Pointer on the AirInv (slave) service handler.

Definition at line 110 of file [AIRINV_Types.hpp](#).

23.1.1.2 typedef boost::shared_ptr<[AIRINV_Master_Service](#)> [AIRINV::AIRINV_Master_ServicePtr_T](#)

(Smart) Pointer on the AirInv master service handler.

Definition at line 115 of file [AIRINV_Types.hpp](#).

23.1.1.3 typedef std::map<const stdair::AirlineCode_T, [AIRINV_ServicePtr_T](#)> [AIRINV::AIRINV_ServicePtr_Map_T](#)

Type defining a map of airline codes and the corresponding airline inventories.

Definition at line 122 of file [AIRINV_Types.hpp](#).

23.1.1.4 typedef std::map<const stdair::DTD_T, double> [AIRINV::FRAT5Curve_T](#)

Define the FRAT5 curve.

Definition at line 127 of file [AIRINV_Types.hpp](#).

23.1.1.5 typedef char [AIRINV::char_t](#)

Definition at line 31 of file [BasParserTypes.hpp](#).

23.1.1.6 typedef boost::spirit::classic::file_iterator<[char_t](#)> [AIRINV::iterator_t](#)

Definition at line 35 of file [BasParserTypes.hpp](#).

23.1.1.7 typedef boost::spirit::classic::scanner<[iterator_t](#)> [AIRINV::scanner_t](#)

Definition at line 36 of file [BasParserTypes.hpp](#).

23.1.1.8 typedef boost::spirit::classic::rule<[scanner_t](#)> [AIRINV::rule_t](#)

Definition at line 37 of file [BasParserTypes.hpp](#).

23.1.1.9 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> [AIRINV::int1_p_t](#)

1-digit-integer parser

Definition at line 45 of file [BasParserTypes.hpp](#).

23.1.1.10 `typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> AIRINV::uint2_p_t`

2-digit-integer parser

Definition at line 48 of file [BasParserTypes.hpp](#).

23.1.1.11 `typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> AIRINV::uint1_2_p_t`

Up-to-2-digit-integer parser

Definition at line 51 of file [BasParserTypes.hpp](#).

23.1.1.12 `typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3> AIRINV::uint1_3_p_t`

Up-to-3-digit-integer parser

Definition at line 54 of file [BasParserTypes.hpp](#).

23.1.1.13 `typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> AIRINV::uint4_p_t`

4-digit-integer parser

Definition at line 57 of file [BasParserTypes.hpp](#).

23.1.1.14 `typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4> AIRINV::uint1_4_p_t`

Up-to-4-digit-integer parser

Definition at line 60 of file [BasParserTypes.hpp](#).

23.1.1.15 `typedef boost::spirit::classic::chset<char_t> AIRINV::chset_t`

character set

Definition at line 63 of file [BasParserTypes.hpp](#).

23.1.1.16 `typedef boost::spirit::classic::impl::loop_traits<chset_t, unsigned int, unsigned int>::type AIRINV::repeat_p_t`

(Repeating) sequence of a given number of characters: `repeat_p(min, max)`

Definition at line 69 of file [BasParserTypes.hpp](#).

23.1.1.17 `typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> AIRINV::bounded2_p_t`

Bounded-number-of-integers parser

Definition at line 72 of file [BasParserTypes.hpp](#).

23.1.1.18 `typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int> AIRINV::bounded1_2_p_t`

Definition at line 73 of file [BasParserTypes.hpp](#).

23.1.1.19 `typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int> AIRINV::bounded1_3_p_t`

Definition at line 74 of file [BasParserTypes.hpp](#).

23.1.1.20 `typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> AIRINV::bounded4_p_t`

Definition at line 75 of file [BasParserTypes.hpp](#).

23.1.1.21 `typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int> AIRINV::bounded1_4_p_t`

Definition at line 76 of file [BasParserTypes.hpp](#).

23.1.1.22 `typedef std::set<stdair::AirportCode_T> AIRINV::AirportList_T`

Define lists of Airport Codes.

Definition at line 16 of file [AirportList.hpp](#).

23.1.1.23 `typedef std::vector<stdair::AirportCode_T> AIRINV::AirportOrderedList_T`

Definition at line 17 of file [AirportList.hpp](#).

23.1.1.24 `typedef std::vector<BookingClassStruct> AIRINV::BookingClassStructList_T`

List of BookingClass structures.

Definition at line 60 of file [BookingClassStruct.hpp](#).

23.1.1.25 `typedef std::vector<BucketStruct> AIRINV::BucketStructList_T`

List of Bucket structures.

Definition at line 44 of file [BucketStruct.hpp](#).

23.1.1.26 `typedef std::vector<FareFamilyStruct> AIRINV::FareFamilyStructList_T`

List of FareFamily-Detail structures.

Definition at line 56 of file [FareFamilyStruct.hpp](#).

23.1.1.27 `typedef std::vector<LegCabinStruct> AIRINV::LegCabinStructList_T`

List of LegCabin-Detail structures.

Definition at line 52 of file [LegCabinStruct.hpp](#).

23.1.1.28 `typedef std::vector<LegStruct> AIRINV::LegStructList_T`

List of Leg structures.

Definition at line 55 of file [LegStruct.hpp](#).

23.1.1.29 `typedef std::vector<SegmentCabinStruct> AIRINV::SegmentCabinStructList_T`

List of SegmentCabin-Detail structures.

Definition at line 48 of file [SegmentCabinStruct.hpp](#).

23.1.1.30 `typedef std::vector<SegmentStruct> AIRINV::SegmentStructList_T`

List of Segment structures.

Definition at line 43 of file [SegmentStruct.hpp](#).

23.1.1.31 `typedef std::map<const stdair::Date_T, stdair::SegmentCabin*> AIRINV::DepartureDateSegmentCabinMap_T`

Definition at line 29 of file [InventoryManager.hpp](#).

23.1.1.32 `typedef std::map<const std::string, DepartureDateSegmentCabinMap_T>
AIRINV::SimilarSegmentCabinSetMap_T`

Definition at line 31 of file [InventoryManager.hpp](#).

23.1.1.33 `typedef boost::shared_ptr<boost::thread> AIRINV::ThreadShrPtr_T`

Definition at line 15 of file [AirInvServer_ASIO.cpp](#).

23.1.1.34 `typedef std::vector<ThreadShrPtr_T> AIRINV::ThreadShrPtrList_T`

Definition at line 16 of file [AirInvServer_ASIO.cpp](#).

23.1.1.35 `typedef boost::shared_ptr<Connection> AIRINV::ConnectionShrPtr_T`

Shared pointer on a [Connection](#) object.

Definition at line 71 of file [Connection.hpp](#).

23.1.2 Variable Documentation

23.1.2.1 `const std::string AIRINV::DEFAULT_AIRLINE_CODE = "BA"`

Default airline name for the [AIRINV_Service](#).

Definition at line 11 of file [BasConst.cpp](#).

23.1.2.2 `const FRAT5Curve_T AIRINV::DEFAULT_PICKUP_FRAT5_CURVE`

Initial value:

```
DefaultMap::createPickupFRAT5Curve()
```

Default pick-up FRAT5 curve for Q-equivalent booking conversion.

Definition at line 14 of file [BasConst.cpp](#).

23.2 AIRINV::DCPParserHelper Namespace Reference

Classes

- struct [ParserSemanticAction](#)
- struct [storeDCPIId](#)
- struct [storeOrigin](#)
- struct [storeDestination](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeStartRangeTime](#)
- struct [storeEndRangeTime](#)
- struct [storePOS](#)
- struct [storeCabinCode](#)
- struct [storeChannel](#)
- struct [storeAdvancePurchase](#)
- struct [storeSaturdayStay](#)
- struct [storeChangeFees](#)
- struct [storeNonRefundable](#)
- struct [storeMinimumStay](#)
- struct [storeDCP](#)
- struct [storeAirlineCode](#)
- struct [storeClass](#)
- struct [doEndDCP](#)
- struct [DCPRuleParser](#)

Variables

- `stdair::int1_p_t` [int1_p](#)
- `stdair::uint2_p_t` [uint2_p](#)
- `stdair::uint4_p_t` [uint4_p](#)
- `stdair::uint1_4_p_t` [uint1_4_p](#)
- `stdair::hour_p_t` [hour_p](#)
- `stdair::minute_p_t` [minute_p](#)
- `stdair::second_p_t` [second_p](#)
- `stdair::year_p_t` [year_p](#)
- `stdair::month_p_t` [month_p](#)
- `stdair::day_p_t` [day_p](#)

23.2.1 Variable Documentation

23.2.1.1 `stdair::int1_p_t` [AIRINV::DCPParserHelper::int1_p](#)

Namespaces. 1-digit-integer parser

Definition at line [427](#) of file [DCPParserHelper.cpp](#).

23.2.1.2 `stdair::uint2_p_t` [AIRINV::DCPParserHelper::uint2_p](#)

2-digit-integer parser

Definition at line [430](#) of file [DCPParserHelper.cpp](#).

23.2.1.3 `stdair::uint4_p_t` [AIRINV::DCPParserHelper::uint4_p](#)

4-digit-integer parser

Definition at line [433](#) of file [DCPParserHelper.cpp](#).

23.2.1.4 `stdair::uint1_4_p_t` [AIRINV::DCPParserHelper::uint1_4_p](#)

Up-to-4-digit-integer parser

Definition at line [436](#) of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.5 `stdair::hour_p_t` [AIRINV::DCPParserHelper::hour_p](#)

Time element parsers.

Definition at line [439](#) of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.6 `stdair::minute_p_t` [AIRINV::DCPParserHelper::minute_p](#)

Definition at line [440](#) of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.7 `stdair::second_p_t` [AIRINV::DCPParserHelper::second_p](#)

Definition at line [441](#) of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.8 `stdair::year_p_t` [AIRINV::DCPParserHelper::year_p](#)

Date element parsers.

Definition at line 444 of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.9 stdair::month_p_t AIRINV::DCPParserHelper::month_p

Definition at line 445 of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.2.1.10 stdair::day_p_t AIRINV::DCPParserHelper::day_p

Definition at line 446 of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser\(\)](#).

23.3 AIRINV::InventoryParserHelper Namespace Reference

Classes

- struct [ParserSemanticAction](#)
- struct [storeSnapshotDate](#)
- struct [storeAirlineCode](#)
- struct [storeFlightNumber](#)
- struct [storeFlightDate](#)
- struct [storeFlightTypeCode](#)
- struct [storeFlightVisibilityCode](#)
- struct [storeLegBoardingPoint](#)
- struct [storeLegOffPoint](#)
- struct [storeBoardingDate](#)
- struct [storeBoardingTime](#)
- struct [storeOffDate](#)
- struct [storeOffTime](#)
- struct [storeLegCabinCode](#)
- struct [storeSaleableCapacity](#)
- struct [storeAU](#)
- struct [storeUPR](#)
- struct [storeBookingCounter](#)
- struct [storeNAV](#)
- struct [storeGAV](#)
- struct [storeACP](#)
- struct [storeETB](#)
- struct [storeYieldUpperRange](#)
- struct [storeBucketAvailability](#)
- struct [storeSeatIndex](#)
- struct [storeSegmentBoardingPoint](#)
- struct [storeSegmentOffPoint](#)
- struct [storeSegmentCabinCode](#)
- struct [storeSegmentCabinBookingCounter](#)
- struct [storeClassCode](#)
- struct [storeSubclassCode](#)
- struct [storeParentClassCode](#)
- struct [storeParentSubclassCode](#)
- struct [storeCumulatedProtection](#)
- struct [storeProtection](#)
- struct [storeNego](#)
- struct [storeNoShow](#)

- struct [storeOverbooking](#)
- struct [storeNbOfBkgs](#)
- struct [storeNbOfGroupBkgs](#)
- struct [storeNbOfPendingGroupBkgs](#)
- struct [storeNbOfStaffBkgs](#)
- struct [storeNbOfWLBkgs](#)
- struct [storeClassETB](#)
- struct [storeClassAvailability](#)
- struct [storeSegmentAvailability](#)
- struct [storeRevenueAvailability](#)
- struct [storeFamilyCode](#)
- struct [storeFCClasses](#)
- struct [doEndFlightDate](#)
- struct [InventoryParser](#)

Functions

- [repeat_p_t airline_code_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#), 2, 3)
- [bounded1_4_p_t flight_number_p](#) ([uint1_4_p](#).[derived\(\)](#), 0u, 9999u)
- [bounded2_p_t year_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 99u)
- [bounded2_p_t month_p](#) ([uint2_p](#).[derived\(\)](#), 1u, 12u)
- [bounded2_p_t day_p](#) ([uint2_p](#).[derived\(\)](#), 1u, 31u)
- [repeat_p_t dow_p](#) ([chset_t](#)("0-1").[derived\(\)](#).[derived\(\)](#), 7, 7)
- [repeat_p_t airport_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#), 3, 3)
- [bounded1_2_p_t hours_p](#) ([uint1_2_p](#).[derived\(\)](#), 0u, 24u)
- [bounded2_p_t minutes_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 59u)
- [bounded2_p_t seconds_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 59u)
- [chset_t cabin_code_p](#) ("A-Z")
- [chset_t class_code_p](#) ("A-Z")
- [chset_t passenger_type_p](#) ("A-Z")
- [repeat_p_t class_code_list_p](#) ([chset_t](#)("A-Z").[derived\(\)](#), 1, 26)
- [bounded1_3_p_t stay_duration_p](#) ([uint1_3_p](#).[derived\(\)](#), 0u, 999u)

Variables

- [int1_p_t int1_p](#)
- [uint2_p_t uint2_p](#)
- [uint1_2_p_t uint1_2_p](#)
- [uint1_3_p_t uint1_3_p](#)
- [uint4_p_t uint4_p](#)
- [uint1_4_p_t uint1_4_p](#)
- [int1_p_t family_code_p](#)

23.3.1 Function Documentation

23.3.1.1 [repeat_p_t AIRINV::InventoryParserHelper::airline_code_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#) , 2 , 3)

Airline Code Parser: [repeat_p](#)(2,3)[[chset_p](#)("0-9A-Z")]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.2 [bounded1_4_p_t AIRINV::InventoryParserHelper::flight_number_p](#) ([uint1_4_p](#).[derived\(\)](#), 0u , 9999u)

Flight Number Parser: [limit_d](#)(0u, 9999u)[[uint1_4_p](#)]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.3 **bounded2_p_t** AIRINV::InventoryParserHelper::year_p (uint2_p. *derived()*, 0u , 99u)

Year Parser: limit_d(00u, 99u)[uint4_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.4 **bounded2_p_t** AIRINV::InventoryParserHelper::month_p (uint2_p. *derived()*, 1u , 12u)

Month Parser: limit_d(1u, 12u)[uint2_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.5 **bounded2_p_t** AIRINV::InventoryParserHelper::day_p (uint2_p. *derived()*, 1u , 31u)

Day Parser: limit_d(1u, 31u)[uint2_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.6 **repeat_p_t** AIRINV::InventoryParserHelper::dow_p (chset_t("0-1").*derived()*.*derived()* , 7 , 7)

DOW (Day-Of-the-Week) Parser: repeat_p(7)[chset_p("0-1")]

23.3.1.7 **repeat_p_t** AIRINV::InventoryParserHelper::airport_p (chset_t("0-9A-Z").*derived()* , 3 , 3)

Airport Parser: repeat_p(3)[chset_p("0-9A-Z")]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.8 **bounded1_2_p_t** AIRINV::InventoryParserHelper::hours_p (uint1_2_p. *derived()*, 0u , 24u)

Hour Parser: limit_d(0u, 24u)[uint2_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.9 **bounded2_p_t** AIRINV::InventoryParserHelper::minutes_p (uint2_p. *derived()*, 0u , 59u)

Minute Parser: limit_d(0u, 59u)[uint2_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.10 **bounded2_p_t** AIRINV::InventoryParserHelper::seconds_p (uint2_p. *derived()*, 0u , 59u)

Second Parser: limit_d(0u, 59u)[uint2_p]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.11 **chset_t** AIRINV::InventoryParserHelper::cabin_code_p ("A-Z")

Cabin code parser: chset_p("A-Z")

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.12 **chset_t** AIRINV::InventoryParserHelper::class_code_p ("A-Z")

Booking class code parser: chset_p("A-Z")

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.13 **chset_t** AIRINV::InventoryParserHelper::passenger_type_p ("A-Z")

Passenger type parser: chset_p("A-Z")

23.3.1.14 **repeat_p_t** AIRINV::InventoryParserHelper::class_code_list_p (chset_t("A-Z").*derived()* , 1 , 26)

Class Code List Parser: repeat_p(1,26)[chset_p("A-Z")]

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.1.15 **bounded1_3_p_t** AIRINV::InventoryParserHelper::stay_duration_p (uint1_3_p. *derived()*, 0u , 999u)

Stay duration Parser: limit_d(0u, 999u)[uint3_p]

23.3.2 Variable Documentation

23.3.2.1 **int1_p_t** AIRINV::InventoryParserHelper::int1_p

1-digit-integer parser

Definition at line 791 of file [InventoryParserHelper.cpp](#).

23.3.2.2 **uint2_p_t** AIRINV::InventoryParserHelper::uint2_p

2-digit-integer parser

Definition at line 794 of file [InventoryParserHelper.cpp](#).

23.3.2.3 **uint1_2_p_t** AIRINV::InventoryParserHelper::uint1_2_p

Up-to-2-digit-integer parser

Definition at line 797 of file [InventoryParserHelper.cpp](#).

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.2.4 **uint1_3_p_t** AIRINV::InventoryParserHelper::uint1_3_p

Up-to-3-digit-integer parser

Definition at line 800 of file [InventoryParserHelper.cpp](#).

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.3.2.5 **uint4_p_t** AIRINV::InventoryParserHelper::uint4_p

4-digit-integer parser

Definition at line 803 of file [InventoryParserHelper.cpp](#).

23.3.2.6 **uint1_4_p_t** AIRINV::InventoryParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 806 of file [InventoryParserHelper.cpp](#).

23.3.2.7 **int1_p_t** AIRINV::InventoryParserHelper::family_code_p

Family code parser

Definition at line 848 of file [InventoryParserHelper.cpp](#).

Referenced by [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition\(\)](#).

23.4 AIRINV::ScheduleParserHelper Namespace Reference

Classes

- struct [ParserSemanticAction](#)
- struct [storeAirlineCode](#)
- struct [storeFlightNumber](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeDow](#)

- struct [storeLegBoardingPoint](#)
- struct [storeLegOffPoint](#)
- struct [storeBoardingTime](#)
- struct [storeOffTime](#)
- struct [storeElapsedTime](#)
- struct [storeLegCabinCode](#)
- struct [storeCapacity](#)
- struct [storeSegmentSpecificity](#)
- struct [storeSegmentBoardingPoint](#)
- struct [storeSegmentOffPoint](#)
- struct [storeSegmentCabinCode](#)
- struct [storeClasses](#)
- struct [storeFamilyCode](#)
- struct [storeFClasses](#)
- struct [doEndFlight](#)
- struct [FlightPeriodParser](#)

Functions

- [repeat_p_t airline_code_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#), 2, 3)
- [bounded1_4_p_t flight_number_p](#) ([uint1_4_p](#).[derived\(\)](#), 0u, 9999u)
- [bounded4_p_t year_p](#) ([uint4_p](#).[derived\(\)](#), 2000u, 2099u)
- [bounded2_p_t month_p](#) ([uint2_p](#).[derived\(\)](#), 1u, 12u)
- [bounded2_p_t day_p](#) ([uint2_p](#).[derived\(\)](#), 1u, 31u)
- [repeat_p_t dow_p](#) ([chset_t](#)("0-1").[derived\(\)](#).[derived\(\)](#), 7, 7)
- [repeat_p_t airport_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#), 3, 3)
- [bounded2_p_t hours_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 23u)
- [bounded2_p_t minutes_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 59u)
- [bounded2_p_t seconds_p](#) ([uint2_p](#).[derived\(\)](#), 0u, 59u)
- [chset_t cabin_code_p](#) ("A-Z")
- [repeat_p_t class_code_list_p](#) ([chset_t](#)("A-Z").[derived\(\)](#), 1, 26)

Variables

- [int1_p_t int1_p](#)
- [uint2_p_t uint2_p](#)
- [uint4_p_t uint4_p](#)
- [uint1_4_p_t uint1_4_p](#)
- [int1_p_t family_code_p](#)

23.4.1 Function Documentation

23.4.1.1 [repeat_p_t AIRINV::ScheduleParserHelper::airline_code_p](#) ([chset_t](#)("0-9A-Z").[derived\(\)](#) , 2 , 3)

Airline Code Parser: [repeat_p](#)(2,3)[[chset_p](#)("0-9A-Z")]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.2 [bounded1_4_p_t AIRINV::ScheduleParserHelper::flight_number_p](#) ([uint1_4_p](#).[derived\(\)](#), 0u , 9999u)

Flight Number Parser: [limit_d](#)(0u, 9999u)[[uint1_4_p](#)]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.3 **bounded4_p_t** AIRINV::ScheduleParserHelper::year_p (uint4_p. *derived()*, 2000u , 2099u)

Year Parser: limit_d(2000u, 2099u)[uint4_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.4 **bounded2_p_t** AIRINV::ScheduleParserHelper::month_p (uint2_p. *derived()*, 1u , 12u)

Month Parser: limit_d(1u, 12u)[uint2_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.5 **bounded2_p_t** AIRINV::ScheduleParserHelper::day_p (uint2_p. *derived()*, 1u , 31u)

Day Parser: limit_d(1u, 31u)[uint2_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.6 **repeat_p_t** AIRINV::ScheduleParserHelper::dow_p (chset_t("0-1").*derived()*.*derived()* , 7 , 7)

DOW (Day-Of-the-Week) Parser: repeat_p(7)[chset_p("0-1")]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.7 **repeat_p_t** AIRINV::ScheduleParserHelper::airport_p (chset_t("0-9A-Z").*derived()* , 3 , 3)

Airport Parser: repeat_p(3)[chset_p("0-9A-Z")]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.8 **bounded2_p_t** AIRINV::ScheduleParserHelper::hours_p (uint2_p. *derived()*, 0u , 23u)

Hour Parser: limit_d(0u, 23u)[uint2_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.9 **bounded2_p_t** AIRINV::ScheduleParserHelper::minutes_p (uint2_p. *derived()*, 0u , 59u)

Minute Parser: limit_d(0u, 59u)[uint2_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.10 **bounded2_p_t** AIRINV::ScheduleParserHelper::seconds_p (uint2_p. *derived()*, 0u , 59u)

Second Parser: limit_d(0u, 59u)[uint2_p]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.11 **chset_t** AIRINV::ScheduleParserHelper::cabin_code_p ("A-Z")

Cabin Code Parser: chset_p("A-Z")

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.1.12 **repeat_p_t** AIRINV::ScheduleParserHelper::class_code_list_p (chset_t("A-Z").*derived()* , 1 , 26)

Class Code List Parser: repeat_p(1,26)[chset_p("A-Z")]

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.2 Variable Documentation

23.4.2.1 **int1_p_t** AIRINV::ScheduleParserHelper::int1_p

1-digit-integer parser

Definition at line 409 of file [ScheduleParserHelper.cpp](#).

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.4.2.2 uint2_p_t AIRINV::ScheduleParserHelper::uint2_p

2-digit-integer parser

Definition at line 412 of file [ScheduleParserHelper.cpp](#).

23.4.2.3 uint4_p_t AIRINV::ScheduleParserHelper::uint4_p

4-digit-integer parser

Definition at line 415 of file [ScheduleParserHelper.cpp](#).

23.4.2.4 uint1_4_p_t AIRINV::ScheduleParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 418 of file [ScheduleParserHelper.cpp](#).

23.4.2.5 int1_p_t AIRINV::ScheduleParserHelper::family_code_p

Family code parser

Definition at line 454 of file [ScheduleParserHelper.cpp](#).

Referenced by [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

23.5 stdair Namespace Reference

Forward declarations.

Classes

- struct [BomPropertyTree](#)

23.5.1 Detailed Description

Forward declarations.

23.6 swift Namespace Reference

The wrapper namespace.

Classes

- class [SKeymap](#)
The readline keymap wrapper.
- class [SReadline](#)
The readline library wrapper.

23.6.1 Detailed Description

The wrapper namespace. The namespace is also used for other library elements.

24 Class Documentation

24.1 AIRINV::AIRINV_Master_Service Class Reference

Interface for the [AIRINV](#) Services.

```
#include <airinv/AIRINV_Master_Service.hpp>
```

Public Member Functions

- [AIRINV_Master_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [AIRINV_Master_Service](#) (const stdair::BasLogParams &)
- [AIRINV_Master_Service](#) (stdair::STDAIR_ServicePtr_T)
- void [parseAndLoad](#) (const stdair::Filename_T &iInventoryFilename)
- void [parseAndLoad](#) (const stdair::Filename_T &iScheduleFilename, const stdair::Filename_T &iODInputFilename, const AIRRAC::YieldFilePath &iYieldFilename)
- [~AIRINV_Master_Service](#) ()
- void [initSnapshotAndRMEvents](#) (const stdair::Date_T &, const stdair::Date_T &)
- void [buildSampleBom](#) ()
- void [calculateAvailability](#) (stdair::TravelSolutionStruct &, const stdair::PartnershipTechnique &)
- bool [sell](#) (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool [cancel](#) (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- void [takeSnapshots](#) (const stdair::SnapshotStruct &)
- void [optimise](#) (const stdair::RMEventStruct &, const stdair::ForecastingMethod &, const stdair::PartnershipTechnique &)
- std::string [jsonExport](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string [list](#) (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- bool [check](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string [csvDisplay](#) () const
- std::string [csvDisplay](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const

24.1.1 Detailed Description

Interface for the [AIRINV](#) Services.

Definition at line 41 of file [AIRINV_Master_Service.hpp](#).

24.1.2 Constructor & Destructor Documentation

24.1.2.1 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (const stdair::BasLogParams & iLogParams, const stdair::BasDBParams & iDBParams)

Constructor.

The `initSlaveAirinvService()` method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that a session can be created on the corresponding database.

Parameters

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
<i>const</i>	stdair::BasDBParams& Parameters for the database access.

Definition at line 44 of file [AIRINV_Master_Service.cpp](#).

24.1.2.2 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (const stdair::BasLogParams & iLogParams)

Constructor.

The initSlaveAirinvService() method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
--------------	---

Definition at line 66 of file [AIRINV_Master_Service.cpp](#).

24.1.2.3 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)

Constructor.

The initSlaveAirinvService() method is called; see the corresponding documentation for more details.

Moreover, as no reference on any output stream is given, it is assumed that the StdAir log service has already been initialised with the proper log output stream by some other methods in the calling chain (for instance, when the [AIRINV_Master_Service](#) is itself being initialised by another library service such as SIMCRS_Service).

Parameters

<i>stdair::STDAIR_ServicePtr_T</i>	Reference on the STDAIR service.
------------------------------------	----------------------------------

Definition at line 87 of file [AIRINV_Master_Service.cpp](#).

24.1.2.4 AIRINV::AIRINV_Master_Service::~~AIRINV_Master_Service ()

Destructor.

Definition at line 103 of file [AIRINV_Master_Service.cpp](#).

24.1.3 Member Function Documentation

24.1.3.1 void AIRINV::AIRINV_Master_Service::parseAndLoad (const stdair::Filename_T & iInventoryFilename)

Parse the inventory dump and load it into memory.

The CSV file, describing the airline inventory for the simulator, is parsed and instantiated in memory accordingly.

Parameters

<i>const</i>	stdair::Filename_T& Filename of the input demand file.
--------------	--

Definition at line 204 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::parseAndLoad\(\)](#).

24.1.3.2 void AIRINV::AIRINV_Master_Service::parseAndLoad (const stdair::Filename_T & iScheduleFilename, const stdair::Filename_T & iODInputFilename, const AIRRAC::YieldFilePath & iYieldFilename)

Parse the schedule and O&D input files, and load them into memory.

The CSV files, describing the airline schedule and the O&Ds for the simulator, are parsed and instantiated in memory accordingly.

Parameters

<i>const</i>	stdair::Filename_T& Filename of the input schedule file.
<i>const</i>	stdair::Filename_T& Filename of the input O&D file.
<i>const</i>	AIRRAC::YieldFilePath& Filename of the input yield file.

Definition at line 227 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::parseAndLoad\(\)](#).

24.1.3.3 void AIRINV::AIRINV_Master_Service::initSnapshotAndRMEvents (const stdair::Date_T & *iStartDate*, const stdair::Date_T & *iEndDate*)

Initialise the snapshot and RM events for the inventories.

Parameters

<i>const</i>	stdair::Date_T& Parameters for the start date.
<i>const</i>	stdair::Date_T& Parameters for the end date.

Definition at line 429 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::initRMEvents\(\)](#).

24.1.3.4 void AIRINV::AIRINV_Master_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the BomRoot instance.

The BOM tree is based on two actual inventories (one for BA, another for AF). Each inventory contains one flight. One of those flights has two legs (and therefore three segments).

Definition at line 252 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::buildSampleBom\(\)](#).

24.1.3.5 void AIRINV::AIRINV_Master_Service::calculateAvailability (stdair::TravelSolutionStruct & *ioTravelSolution*, const stdair::PartnershipTechnique & *iPartnershipTechnique*)

Compute the availability for the given travel solution.

Definition at line 468 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::calculateAvailability\(\)](#).

24.1.3.6 bool AIRINV::AIRINV_Master_Service::sell (const std::string & *iSegmentDateKey*, const stdair::ClassCode_T & *iClassCode*, const stdair::PartySize_T & *iPartySize*)

Register a booking.

Parameters

<i>const</i>	std::string& Key for the segment on which the sale is made.
<i>const</i>	stdair::ClassCode_T& Class code where the sale is made.
<i>const</i>	stdair::PartySize_T& Party size.

Returns

bool Whether or not the sale was successfull

Definition at line 499 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::sell\(\)](#).

24.1.3.7 bool [AIRINV::AIRINV_Master_Service::cancel](#) (const std::string & *iSegmentDateKey*, const stdair::ClassCode_T & *iClassCode*, const stdair::PartySize_T & *iPartySize*)

Register a cancellation.

Parameters

<i>const</i>	std::string& Key for the segment on which the cancellation is made.
<i>const</i>	stdair::ClassCode_T& Class code where the sale is made.
<i>const</i>	stdair::PartySize_T& Party size.

Returns

bool Whether or not the sale was successfull

Definition at line 541 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::cancel\(\)](#).

24.1.3.8 void [AIRINV::AIRINV_Master_Service::takeSnapshots](#) (const stdair::SnapshotStruct & *iSnapshot*)

Take inventory snapshots.

Definition at line 584 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::takeSnapshots\(\)](#).

24.1.3.9 void [AIRINV::AIRINV_Master_Service::optimise](#) (const stdair::RMEventStruct & *iRMEvent*, const stdair::ForecastingMethod & *iForecastingMethod*, const stdair::PartnershipTechnique & *iPartnershipTechnique*)

Optimise (revenue management) an flight-date/network-date

Definition at line 610 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::optimise\(\)](#).

24.1.3.10 std::string [AIRINV::AIRINV_Master_Service::jsonExport](#) (const stdair::AirlineCode_T & *iAirlineCode*, const stdair::FlightNumber_T & *iFlightNumber*, const stdair::Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to dump.
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to dump.
<i>const</i>	stdair::Date_T& Departure date of the flight to dump.

Returns

std::string Output string in which the BOM tree is JSON-ified.

Definition at line 304 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::jsonExport\(\)](#).

24.1.3.11 `std::string AIRINV::AIRINV_Master_Service::list (const stdair::AirlineCode_T & iAirlineCode = "all",
const stdair::FlightNumber_T & iFlightNumber = 0) const`

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters

<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 330 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::list\(\)](#).

24.1.3.12 `bool AIRINV::AIRINV_Master_Service::check (const stdair::AirlineCode_T & iAirlineCode, const
stdair::FlightNumber_T & iFlightNumber, const stdair::Date_T & iDepartureDate) const`

Check whether the given flight-date is a valid one.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to check.
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to check.
<i>const</i>	stdair::Date_T& Departure date of the flight to check.

Returns

bool Whether or not the given flight date is valid.

Definition at line 355 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::check\(\)](#).

24.1.3.13 `std::string AIRINV::AIRINV_Master_Service::csvDisplay () const`

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 380 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::csvDisplay\(\)](#).

24.1.3.14 `std::string AIRINV::AIRINV_Master_Service::csvDisplay (const stdair::AirlineCode_T & iAirlineCode, const
stdair::FlightNumber_T & iFlightNumber, const stdair::Date_T & iDepartureDate) const`

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to display.
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to display.
<i>const</i>	stdair::Date_T& Departure date of the flight to display.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 403 of file [AIRINV_Master_Service.cpp](#).

References [AIRINV::AIRINV_Service::csvDisplay\(\)](#).

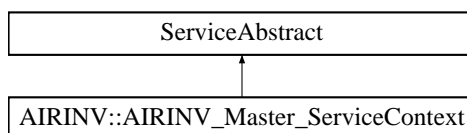
The documentation for this class was generated from the following files:

- [airinv/AIRINV_Master_Service.hpp](#)
- [airinv/service/AIRINV_Master_Service.cpp](#)

24.2 AIRINV::AIRINV_Master_ServiceContext Class Reference

```
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
```

Inheritance diagram for AIRINV::AIRINV_Master_ServiceContext:

**Friends**

- class [AIRINV_Master_Service](#)
- class [FacAirinvMasterServiceContext](#)

24.2.1 Detailed Description

Class holding the context of the Airinv services.

Definition at line 26 of file [AIRINV_Master_ServiceContext.hpp](#).

24.2.2 Friends And Related Function Documentation**24.2.2.1 friend class AIRINV_Master_Service [friend]**

The [AIRINV_Master_Service](#) class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 32 of file [AIRINV_Master_ServiceContext.hpp](#).

24.2.2.2 friend class FacAirinvMasterServiceContext [friend]

Definition at line 33 of file [AIRINV_Master_ServiceContext.hpp](#).

The documentation for this class was generated from the following files:

- [airinv/service/AIRINV_Master_ServiceContext.hpp](#)
- [airinv/service/AIRINV_Master_ServiceContext.cpp](#)

24.3 AIRINV::AIRINV_Service Class Reference

Interface for the [AIRINV](#) Services.

```
#include <airinv/AIRINV_Service.hpp>
```

Public Member Functions

- [AIRINV_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [AIRINV_Service](#) (const stdair::BasLogParams &)
- [AIRINV_Service](#) (stdair::STDAIR_ServicePtr_T)
- void [parseAndLoad](#) (const stdair::Filename_T &iInventoryFilename)
- void [parseAndLoad](#) (const stdair::Filename_T &iScheduleFilename, const stdair::Filename_T &iODInputFilename, const AIRRAC::YieldFilePath &iYieldFilename)
- [~AIRINV_Service](#) ()
- void [buildSampleBom](#) ()
- stdair::RMEventList_T [initRMEvents](#) (const stdair::Date_T &iStartDate, const stdair::Date_T &iEndDate)
- void [calculateAvailability](#) (stdair::TravelSolutionStruct &, const stdair::PartnershipTechnique &)
- bool [sell](#) (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool [cancel](#) (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- void [takeSnapshots](#) (const stdair::AirlineCode_T &, const stdair::DateTime_T &)
- void [optimise](#) (const stdair::AirlineCode_T &, const stdair::KeyDescription_T &, const stdair::DateTime_T &, const stdair::ForecastingMethod &, const stdair::PartnershipTechnique &)
- std::string [jsonExport](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string [list](#) (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- bool [check](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string [csvDisplay](#) () const
- std::string [csvDisplay](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const

24.3.1 Detailed Description

Interface for the [AIRINV](#) Services.

Definition at line 37 of file [AIRINV_Service.hpp](#).

24.3.2 Constructor & Destructor Documentation

24.3.2.1 AIRINV::AIRINV_Service::AIRINV_Service (const stdair::BasLogParams & iLogParams, const stdair::BasDBParams & iDBParams)

Constructor.

The `initAirinvService()` method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that a session can be created on the corresponding database.

Parameters

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
<i>const</i>	stdair::BasDBParams& Parameters for the database access.

Definition at line 74 of file [AIRINV_Service.cpp](#).

24.3.2.2 AIRINV::AIRINV_Service::AIRINV_Service (const stdair::BasLogParams & iLogParams)

Constructor.

The `initAirinvService()` method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
--------------	---

Definition at line 48 of file [AIRINV_Service.cpp](#).

24.3.2.3 AIRINV::AIRINV_Service::AIRINV_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)

Constructor.

The initAirinvService() method is called; see the corresponding documentation for more details.

Moreover, as no reference on any output stream is given, it is assumed that the StdAir log service has already been initialised with the proper log output stream by some other methods in the calling chain (for instance, when the [AIRINV_Master_Service](#) is itself being initialised by another library service such as SIMCRS_Service).

Parameters

<i>stdair::STDAIR_ServicePtr_T</i>	Reference on the STDAIR service.
<i>const</i>	stdair::Filename_T& Filename of the input inventory file.

Definition at line 101 of file [AIRINV_Service.cpp](#).

24.3.2.4 AIRINV::AIRINV_Service::~~AIRINV_Service ()

Destructor.

Definition at line 124 of file [AIRINV_Service.cpp](#).

24.3.3 Member Function Documentation

24.3.3.1 void AIRINV::AIRINV_Service::parseAndLoad (const stdair::Filename_T & iInventoryFilename)

Parse the inventory dump and load it into memory.

The CSV file, describing the airline inventory for the simulator, is parsed and instantiated in memory accordingly.

Parameters

<i>const</i>	stdair::Filename_T& Filename of the input demand file.
--------------	--

Definition at line 251 of file [AIRINV_Service.cpp](#).

References [AIRINV::InventoryParser::buildInventory\(\)](#).

Referenced by [AIRINV::AIRINV_Master_Service::parseAndLoad\(\)](#).

24.3.3.2 void AIRINV::AIRINV_Service::parseAndLoad (const stdair::Filename_T & iScheduleFilename, const stdair::Filename_T & iODInputFilename, const AIRRAC::YieldFilePath & iYieldFilename)

Parse the schedule and O&D input files, and load them into memory.

The CSV files, describing the airline schedule and the O&Ds for the simulator, are parsed and instantiated in memory accordingly.

Parameters

<i>const</i>	stdair::Filename_T& Filename of the input schedule file.
<i>const</i>	stdair::Filename_T& Filename of the input O&D file.
<i>const</i>	AIRRAC::YieldFilePath& Filename of the input yield file.

Definition at line 266 of file [AIRINV_Service.cpp](#).

References [AIRINV::ScheduleParser::generateInventories\(\)](#).

24.3.3.3 void AIRINV::AIRINV_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the BomRoot instance.

The BOM tree is based on two actual inventories (one for BA, another for AF). Each inventory contains one flight. One of those flights has two legs (and therefore three segments).

Definition at line 290 of file [AIRINV_Service.cpp](#).

References [AIRINV::InventoryManager::buildSimilarSegmentCabinSets\(\)](#).

Referenced by [AIRINV::AIRINV_Master_Service::buildSampleBom\(\)](#).

24.3.3.4 stdair::RMEventList_T AIRINV::AIRINV_Service::initRMEvents (const stdair::Date_T & iStartDate, const stdair::Date_T & iEndDate)

Initialise the RM events for the inventory.

Parameters

<i>const</i>	stdair::Date_T& Parameters for the start date.
<i>const</i>	stdair::Date_T& Parameters for the end date.

Definition at line 493 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::initSnapshotAndRMEvents\(\)](#).

24.3.3.5 void AIRINV::AIRINV_Service::calculateAvailability (stdair::TravelSolutionStruct & ioTravelSolution, const stdair::PartnershipTechnique & iPartnershipTechnique)

Compute the availability for the given travel solution.

Definition at line 525 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::calculateAvailability\(\)](#).

24.3.3.6 bool AIRINV::AIRINV_Service::sell (const std::string & iSegmentDateKey, const stdair::ClassCode_T & iClassCode, const stdair::PartySize_T & iPartySize)

Register a booking.

Parameters

<i>const</i>	std::string& Key for the segment on which the sale is made
<i>const</i>	stdair::ClassCode_T& Class code where the sale is made
<i>const</i>	stdair::PartySize_T& Party size

Returns

bool Whether or not the sale was successful

Definition at line 552 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::sell\(\)](#).

24.3.3.7 bool AIRINV::AIRINV_Service::cancel (const std::string & iSegmentDateKey, const stdair::ClassCode_T & iClassCode, const stdair::PartySize_T & iPartySize)

Register a cancellation.

Parameters

<i>const</i>	std::string& Key for the segment on which the cancellation is made
<i>const</i>	stdair::ClassCode_T& Class code where the sale is made
<i>const</i>	stdair::PartySize_T& Party size

Returns

bool Whether or not the sale was successfull

Definition at line 593 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::cancel\(\)](#).

24.3.3.8 void **AIRINV::AIRINV_Service::takeSnapshots** (const stdair::AirlineCode_T & *iAirlineCode*, const stdair::DateTime_T & *iSnapshotTime*)

Take inventory snapshots.

Definition at line 637 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::takeSnapshots\(\)](#).

24.3.3.9 void **AIRINV::AIRINV_Service::optimise** (const stdair::AirlineCode_T & *iAirlineCode*, const stdair::KeyDescription_T & *iFDDescription*, const stdair::DateTime_T & *iRMEventTime*, const stdair::ForecastingMethod & *iForecastingMethod*, const stdair::PartnershipTechnique & *iPartnershipTechnique*)

Optimise (revenue management) an flight-date/network-date

Definition at line 664 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::optimise\(\)](#).

24.3.3.10 std::string **AIRINV::AIRINV_Service::jsonExport** (const stdair::AirlineCode_T & *iAirlineCode*, const stdair::FlightNumber_T & *iFlightNumber*, const stdair::Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to dump.
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to dump.
<i>const</i>	stdair::Date_T& Departure date of the flight to dump.

Returns

std::string Output string in which the BOM tree is JSON-ified.

Definition at line 376 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::jsonExport\(\)](#).

24.3.3.11 std::string **AIRINV::AIRINV_Service::list** (const stdair::AirlineCode_T & *iAirlineCode* = "all", const stdair::FlightNumber_T & *iFlightNumber* = 0) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters

<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 400 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::list\(\)](#).

24.3.3.12 `bool AIRINV::AIRINV_Service::check (const stdair::AirlineCode_T & iAirlineCode, const stdair::FlightNumber_T & iFlightNumber, const stdair::Date_T & iDepartureDate) const`

Check whether the given flight-date is a valid one.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to check.
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to check.
<i>const</i>	stdair::Date_T& Departure date of the flight to check.

Returns

bool Whether or not the given flight date is valid.

Definition at line 424 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::check\(\)](#).

24.3.3.13 `std::string AIRINV::AIRINV_Service::csvDisplay () const`

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 448 of file [AIRINV_Service.cpp](#).

Referenced by [AIRINV::AIRINV_Master_Service::csvDisplay\(\)](#).

24.3.3.14 `std::string AIRINV::AIRINV_Service::csvDisplay (const stdair::AirlineCode_T & iAirlineCode, const stdair::FlightNumber_T & iFlightNumber, const stdair::Date_T & iDepartureDate) const`

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the flight to display
<i>const</i>	stdair::FlightNumber_T& Flight number of the flight to display.
<i>const</i>	stdair::Date_T& Departure date of the flight to display.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 469 of file [AIRINV_Service.cpp](#).

The documentation for this class was generated from the following files:

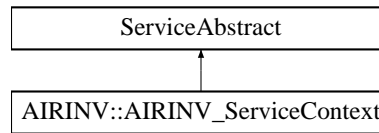
- [airinv/AIRINV_Service.hpp](#)
- [airinv/service/AIRINV_Service.cpp](#)

24.4 AIRINV::AIRINV_ServiceContext Class Reference

Class holding the context of the AirInv services.

```
#include <airinv/service/AIRINV_ServiceContext.hpp>
```

Inheritance diagram for AIRINV::AIRINV_ServiceContext:



Friends

- class [AIRINV_Service](#)
- class [FacAirinvServiceContext](#)

24.4.1 Detailed Description

Class holding the context of the AirInv services.

Definition at line 26 of file [AIRINV_ServiceContext.hpp](#).

24.4.2 Friends And Related Function Documentation

24.4.2.1 friend class AIRINV_Service [friend]

The [AIRINV_Service](#) class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 32 of file [AIRINV_ServiceContext.hpp](#).

24.4.2.2 friend class FacAirinvServiceContext [friend]

Definition at line 33 of file [AIRINV_ServiceContext.hpp](#).

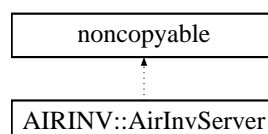
The documentation for this class was generated from the following files:

- [airinv/service/AIRINV_ServiceContext.hpp](#)
- [airinv/service/AIRINV_ServiceContext.cpp](#)

24.5 AIRINV::AirInvServer Class Reference

```
#include <airinv/server/AirInvServer.hpp>
```

Inheritance diagram for AIRINV::AirInvServer:



Public Member Functions

- [AirInvServer](#) (const std::string &address, const std::string &port, const stdair::AirlineCode_T &iAirlineCode, std::size_t thread_pool_size)
- [~AirInvServer](#) ()
- void [run](#) ()
- void [stop](#) ()

24.5.1 Detailed Description

The top-level class of the AirInv server.

Definition at line 23 of file [AirInvServer.hpp](#).

24.5.2 Constructor & Destructor Documentation

24.5.2.1 AIRINV::AirInvServer::AirInvServer (const std::string & address, const std::string & port, const stdair::AirlineCode_T & iAirlineCode, std::size_t thread_pool_size)

Constructor.

Construct the server to listen on the specified TCP address and port, and serve up files from the given directory.

Definition at line 20 of file [AirInvServer_ASIO.cpp](#).

24.5.2.2 AIRINV::AirInvServer::~~AirInvServer ()

Destructor.

Definition at line 46 of file [AirInvServer_ASIO.cpp](#).

24.5.3 Member Function Documentation

24.5.3.1 void AIRINV::AirInvServer::run ()

Run the server's io_service loop.

Definition at line 50 of file [AirInvServer_ASIO.cpp](#).

Referenced by [main\(\)](#).

24.5.3.2 void AIRINV::AirInvServer::stop ()

Stop the server.

Definition at line 69 of file [AirInvServer_ASIO.cpp](#).

The documentation for this class was generated from the following files:

- [airinv/server/AirInvServer.hpp](#)
- [airinv/server/AirInvServer_ASIO.cpp](#)

24.6 AIRINV::BomAbstract Class Reference

```
#include <airinv/bom/BomAbstract.hpp>
```

Public Member Functions

- virtual void [toStream](#) (std::ostream &ioOut) const =0

- virtual void [fromStream](#) (std::istream &ioln)=0
- virtual std::string [toString](#) () const =0
- virtual std::string [describeKey](#) () const =0
- virtual std::string [describeShortKey](#) () const =0

Protected Member Functions

- [BomAbstract](#) ()
- [BomAbstract](#) (const [BomAbstract](#) &)
- virtual [~BomAbstract](#) ()

Friends

- class [FacBomAbstract](#)

24.6.1 Detailed Description

Base class for the Business Object Model (BOM) layer.

Definition at line 14 of file [BomAbstract.hpp](#).

24.6.2 Constructor & Destructor Documentation

24.6.2.1 AIRINV::BomAbstract::BomAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 40 of file [BomAbstract.hpp](#).

24.6.2.2 AIRINV::BomAbstract::BomAbstract (const BomAbstract &) [inline, protected]

Definition at line 41 of file [BomAbstract.hpp](#).

24.6.2.3 virtual AIRINV::BomAbstract::~~BomAbstract () [inline, protected, virtual]

Destructor.

Definition at line 44 of file [BomAbstract.hpp](#).

24.6.3 Member Function Documentation

24.6.3.1 virtual void AIRINV::BomAbstract::toStream (std::ostream & ioOut) const [pure virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

24.6.3.2 virtual void AIRINV::BomAbstract::fromStream (std::istream & ioln) [pure virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Referenced by [operator>>\(\)](#).

24.6.3.3 virtual std::string AIRINV::BomAbstract::toString () const [pure virtual]

Get the serialised version of the Business Object.

24.6.3.4 virtual std::string AIRINV::BomAbstract::describeKey () const [pure virtual]

Get a string describing the whole key (differentiating two objects at any level).

24.6.3.5 virtual std::string AIRINV::BomAbstract::describeShortKey () const [pure virtual]

Get a string describing the short key (differentiating two objects at the same level).

24.6.4 Friends And Related Function Documentation

24.6.4.1 friend class FacBomAbstract [friend]

Definition at line 15 of file [BomAbstract.hpp](#).

The documentation for this class was generated from the following file:

- [airinv/bom/BomAbstract.hpp](#)

24.7 stdair::BomPropertyTree Struct Reference

```
#include <airinv/server/BomPropertyTree.hpp>
```

Public Member Functions

- void [load](#) (const std::string &iBomTree)
- std::string [save](#) () const

Public Attributes

- stdair::AirlineCode_T [_airlineCode](#)
- stdair::FlightNumber_T [_flightNumber](#)
- stdair::Date_T [_departureDate](#)
- std::set< stdair::AirportCode_T > [_airportCodeList](#)

24.7.1 Detailed Description

Structure representing a list of airports.

Definition at line 19 of file [BomPropertyTree.hpp](#).

24.7.2 Member Function Documentation

24.7.2.1 void stdair::BomPropertyTree::load (const std::string & iBomTree)

Update the current BOM tree (*this) with the parsed stream, which is JSON formatted.

Definition at line 17 of file [BomPropertyTree.cpp](#).

References [_airlineCode](#), [_departureDate](#), and [_flightNumber](#).

24.7.2.2 std::string stdair::BomPropertyTree::save () const

Dump the BOM tree (*this) into the stream with a JSON format.

Definition at line 60 of file [BomPropertyTree.cpp](#).

References [_airlineCode](#), [_airportCodeList](#), [_departureDate](#), and [_flightNumber](#).

24.7.3 Member Data Documentation

24.7.3.1 stdair::AirlineCode_T stdair::BomPropertyTree::_airlineCode

Airline code.

Definition at line 33 of file [BomPropertyTree.hpp](#).

Referenced by [load\(\)](#), and [save\(\)](#).

24.7.3.2 stdair::FlightNumber_T stdair::BomPropertyTree::_flightNumber

Flight number.

Definition at line 36 of file [BomPropertyTree.hpp](#).

Referenced by [load\(\)](#), and [save\(\)](#).

24.7.3.3 stdair::Date_T stdair::BomPropertyTree::_departureDate

Departure date.

Definition at line 39 of file [BomPropertyTree.hpp](#).

Referenced by [load\(\)](#), and [save\(\)](#).

24.7.3.4 std::set<stdair::AirportCode_T> stdair::BomPropertyTree::_airportCodeList

Just to have a list, for now.

Definition at line 42 of file [BomPropertyTree.hpp](#).

Referenced by [save\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/server/BomPropertyTree.hpp](#)
- [airinv/server/BomPropertyTree.cpp](#)

24.8 AIRINV::BomRootHelper Class Reference

```
#include <airinv/bom/BomRootHelper.hpp>
```

Static Public Member Functions

- static void [fillFromRouting](#) (const stdair::BomRoot &)

24.8.1 Detailed Description

Class representing the actual business functions for an airline bom root.

Definition at line 16 of file [BomRootHelper.hpp](#).

24.8.2 Member Function Documentation

24.8.2.1 void AIRINV::BomRootHelper::fillFromRouting (const stdair::BomRoot & iBomRoot) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 16 of file [BomRootHelper.cpp](#).

Referenced by [AIRINV::InventoryManager::createDirectAccesses\(\)](#).

The documentation for this class was generated from the following files:

- [airinv/bom/BomRootHelper.hpp](#)
- [airinv/bom/BomRootHelper.cpp](#)

24.9 AIRINV::BookingClassHelper Class Reference

```
#include <airinv/bom/BookingClassHelper.hpp>
```

24.9.1 Detailed Description

Class representing the actual business functions for an airline booking class.

Definition at line 19 of file [BookingClassHelper.hpp](#).

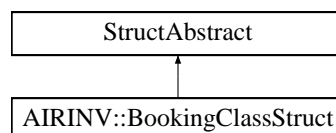
The documentation for this class was generated from the following file:

- [airinv/bom/BookingClassHelper.hpp](#)

24.10 AIRINV::BookingClassStruct Struct Reference

```
#include <airinv/bom/BookingClassStruct.hpp>
```

Inheritance diagram for AIRINV::BookingClassStruct:



Public Member Functions

- stdair::ClassCode_T [getFullSubclassCode](#) () const
- void [fill](#) (stdair::BookingClass &) const
- const std::string [describe](#) () const
- [BookingClassStruct](#) ()

Public Attributes

- stdair::ClassCode_T [_classCode](#)
- stdair::SubclassCode_T [_subclassCode](#)
- stdair::ClassCode_T [_parentClassCode](#)
- stdair::SubclassCode_T [_parentSubclassCode](#)
- stdair::AuthorizationLevel_T [_cumulatedProtection](#)
- stdair::AuthorizationLevel_T [_protection](#)

- [stdair::NbOfSeats_T _nego](#)
- [stdair::OverbookingRate_T _noShowPercentage](#)
- [stdair::OverbookingRate_T _overbookingPercentage](#)
- [stdair::NbOfBookings_T _nbOfBookings](#)
- [stdair::NbOfBookings_T _nbOfGroupBookings](#)
- [stdair::NbOfBookings_T _nbOfPendingGroupBookings](#)
- [stdair::NbOfBookings_T _nbOfStaffBookings](#)
- [stdair::NbOfBookings_T _nbOfWLBookings](#)
- [stdair::NbOfBookings_T _etb](#)
- [stdair::Availability_T _netClassAvailability](#)
- [stdair::Availability_T _segmentAvailability](#)
- [stdair::Availability_T _netRevenueAvailability](#)

24.10.1 Detailed Description

Utility Structure for the parsing of BookingClass structures.

Definition at line 24 of file [BookingClassStruct.hpp](#).

24.10.2 Constructor & Destructor Documentation

24.10.2.1 AIRINV::BookingClassStruct::BookingClassStruct ()

Default Constructor.

Definition at line 16 of file [BookingClassStruct.cpp](#).

24.10.3 Member Function Documentation

24.10.3.1 stdair::ClassCode_T AIRINV::BookingClassStruct::getFullSubclassCode () const

Returns the concatenation of the class and subclass codes.

Definition at line 20 of file [BookingClassStruct.cpp](#).

References [_classCode](#), and [_subclassCode](#).

24.10.3.2 void AIRINV::BookingClassStruct::fill (stdair::BookingClass & ioBookingClass) const

Fill the BookingClass objects with the attributes of the [BookingClassStruct](#).

Definition at line 44 of file [BookingClassStruct.cpp](#).

24.10.3.3 const std::string AIRINV::BookingClassStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 27 of file [BookingClassStruct.cpp](#).

References [_classCode](#), [_cumulatedProtection](#), [_etb](#), [_nbOfBookings](#), [_nbOfGroupBookings](#), [_nbOfPendingGroupBookings](#), [_nbOfStaffBookings](#), [_nbOfWLBookings](#), [_nego](#), [_netClassAvailability](#), [_netRevenueAvailability](#), [_noShowPercentage](#), [_overbookingPercentage](#), [_parentClassCode](#), [_parentSubclassCode](#), [_protection](#), [_segmentAvailability](#), and [_subclassCode](#).

Referenced by [AIRINV::FareFamilyStruct::describe\(\)](#).

24.10.4 Member Data Documentation

24.10.4.1 stdair::ClassCode_T AIRINV::BookingClassStruct::_classCode

Definition at line 26 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), [getFullSubclassCode\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#)(), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#)().

24.10.4.2 stdair::SubclassCode_T AIRINV::BookingClassStruct::_subclassCode

Definition at line 27 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), [getFullSubclassCode\(\)](#), and [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#)().

24.10.4.3 stdair::ClassCode_T AIRINV::BookingClassStruct::_parentClassCode

Definition at line 28 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#)().

24.10.4.4 stdair::SubclassCode_T AIRINV::BookingClassStruct::_parentSubclassCode

Definition at line 29 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#)().

24.10.4.5 stdair::AuthorizationLevel_T AIRINV::BookingClassStruct::_cumulatedProtection

Definition at line 30 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#)().

24.10.4.6 stdair::AuthorizationLevel_T AIRINV::BookingClassStruct::_protection

Definition at line 31 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#)().

24.10.4.7 stdair::NbOfSeats_T AIRINV::BookingClassStruct::_nego

Definition at line 32 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#)().

24.10.4.8 stdair::OverbookingRate_T AIRINV::BookingClassStruct::_noShowPercentage

Definition at line 33 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#)().

24.10.4.9 stdair::OverbookingRate_T AIRINV::BookingClassStruct::_overbookingPercentage

Definition at line 34 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#)().

24.10.4.10 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfBookings

Definition at line 35 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#)().

24.10.4.11 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfGroupBookings

Definition at line 36 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#).

24.10.4.12 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfPendingGroupBookings

Definition at line 37 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#).

24.10.4.13 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfStaffBookings

Definition at line 38 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#).

24.10.4.14 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfWLBookings

Definition at line 39 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#).

24.10.4.15 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_etb

Definition at line 40 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#).

24.10.4.16 stdair::Availability_T AIRINV::BookingClassStruct::_netClassAvailability

Definition at line 41 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#).

24.10.4.17 stdair::Availability_T AIRINV::BookingClassStruct::_segmentAvailability

Definition at line 42 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#).

24.10.4.18 stdair::Availability_T AIRINV::BookingClassStruct::_netRevenueAvailability

Definition at line 43 of file [BookingClassStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#).

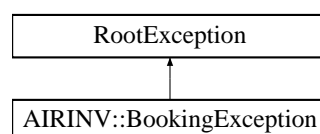
The documentation for this struct was generated from the following files:

- [airinv/bom/BookingClassStruct.hpp](#)
- [airinv/bom/BookingClassStruct.cpp](#)

24.11 AIRINV::BookingException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::BookingException:



24.11.1 Detailed Description

Specific exception related to bookings made against the inventory.

Definition at line 102 of file [AIRINV_Types.hpp](#).

The documentation for this class was generated from the following file:

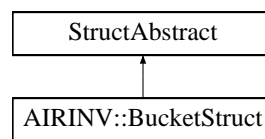
- [airinv/AIRINV_Types.hpp](#)

24.12 AIRINV::BucketStruct Struct Reference

Utility Structure for the parsing of Bucket structures.

```
#include <airinv/bom/BucketStruct.hpp>
```

Inheritance diagram for AIRINV::BucketStruct:



Public Member Functions

- void [fill](#) (stdair::Bucket &) const
- const std::string [describe](#) () const
- [BucketStruct](#) ()

Public Attributes

- stdair::Yield_T [_yieldRangeUpperValue](#)
- stdair::CabinCapacity_T [_availability](#)
- stdair::NbOfSeats_T [_nbOfSeats](#)
- stdair::SeatIndex_T [_seatIndex](#)

24.12.1 Detailed Description

Utility Structure for the parsing of Bucket structures.

Definition at line 26 of file [BucketStruct.hpp](#).

24.12.2 Constructor & Destructor Documentation

24.12.2.1 AIRINV::BucketStruct::BucketStruct ()

Default Constructor.

Definition at line 16 of file [BucketStruct.cpp](#).

24.12.3 Member Function Documentation

24.12.3.1 void AIRINV::BucketStruct::fill (stdair::Bucket & *ioBucket*) const

Fill the Bucket objects with the attributes of the [BucketStruct](#).

Definition at line 29 of file [BucketStruct.cpp](#).

References [_availability](#), [_nbOfSeats](#), and [_yieldRangeUpperValue](#).

24.12.3.2 `const std::string AIRINV::BucketStruct::describe () const`

Give a description of the structure (for display purposes).

Definition at line 20 of file [BucketStruct.cpp](#).

References [_availability](#), [_nbOfSeats](#), [_seatIndex](#), and [_yieldRangeUpperValue](#).

Referenced by [AIRINV::LegCabinStruct::describe\(\)](#).

24.12.4 Member Data Documentation

24.12.4.1 `stdair::Yield_T AIRINV::BucketStruct::_yieldRangeUpperValue`

Definition at line 28 of file [BucketStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#).

24.12.4.2 `stdair::CabinCapacity_T AIRINV::BucketStruct::_availability`

Definition at line 29 of file [BucketStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#).

24.12.4.3 `stdair::NbOfSeats_T AIRINV::BucketStruct::_nbOfSeats`

Definition at line 30 of file [BucketStruct.hpp](#).

Referenced by [describe\(\)](#), and [fill\(\)](#).

24.12.4.4 `stdair::SeatIndex_T AIRINV::BucketStruct::_seatIndex`

Definition at line 31 of file [BucketStruct.hpp](#).

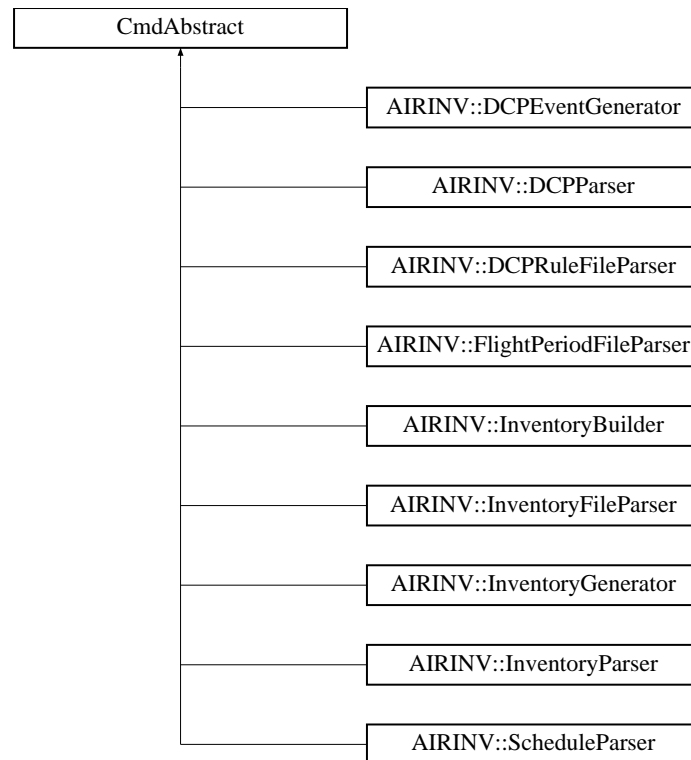
Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/bom/BucketStruct.hpp](#)
- [airinv/bom/BucketStruct.cpp](#)

24.13 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



The documentation for this class was generated from the following file:

- [airinv/command/InventoryBuilder.hpp](#)

24.14 COMMAND Struct Reference

```
#include <airinv/ui/cmdline/readline_autocomp.hpp>
```

Public Attributes

- `char const * name`
- `pt2Func * func`
- `char * doc`

24.14.1 Detailed Description

A structure which contains information on the commands this program can understand.

Definition at line 41 of file [readline_autocomp.hpp](#).

24.14.2 Member Data Documentation

24.14.2.1 `char const* COMMAND::name`

User printable name of the function.

Definition at line 45 of file [readline_autocomp.hpp](#).

Referenced by [com_help\(\)](#), and [find_command\(\)](#).

24.14.2.2 pt2Func* COMMAND::func

Function to call to do the job.

Definition at line 50 of file [readline_autocomp.hpp](#).

Referenced by [execute_line\(\)](#).

24.14.2.3 char* COMMAND::doc

Documentation for this function.

Definition at line 55 of file [readline_autocomp.hpp](#).

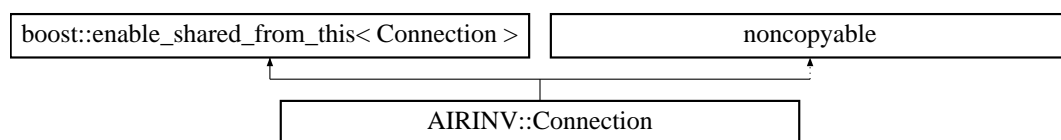
The documentation for this struct was generated from the following file:

- [airinv/ui/cmdline/readline_autocomp.hpp](#)

24.15 AIRINV::Connection Class Reference

```
#include <airinv/server/Connection.hpp>
```

Inheritance diagram for AIRINV::Connection:



Public Member Functions

- [Connection](#) (boost::asio::io_service &, [RequestHandler](#) &)
- boost::asio::ip::tcp::socket & [socket](#) ()
- void [start](#) ()

24.15.1 Detailed Description

Represents a single connection from a client.

Definition at line 25 of file [Connection.hpp](#).

24.15.2 Constructor & Destructor Documentation

24.15.2.1 AIRINV::Connection::Connection (boost::asio::io_service & ioService, RequestHandler & ioHandler)

Constructor.

Construct a connection with the given io_service.

Definition at line 16 of file [Connection.cpp](#).

24.15.3 Member Function Documentation

24.15.3.1 boost::asio::ip::tcp::socket & AIRINV::Connection::socket ()

Get the socket associated with the connection.

Definition at line 22 of file [Connection.cpp](#).

24.15.3.2 void AIRINV::Connection::start ()

Start the first asynchronous operation for the connection.

Definition at line 27 of file [Connection.cpp](#).

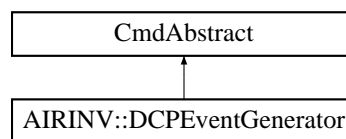
The documentation for this class was generated from the following files:

- [airinv/server/Connection.hpp](#)
- [airinv/server/Connection.cpp](#)

24.16 AIRINV::DCPEventGenerator Class Reference

```
#include <airinv/command/vault/DCPEventGenerator.hpp>
```

Inheritance diagram for AIRINV::DCPEventGenerator:



Friends

- class [DCPFileParser](#)
- struct [DCPParserHelper::doEndDCP](#)
- class [DCPParser](#)

24.16.1 Detailed Description

Class handling the generation / instantiation of the DCP BOM.

Definition at line 27 of file [DCPEventGenerator.hpp](#).

24.16.2 Friends And Related Function Documentation

24.16.2.1 friend class DCPFileParser [friend]

Definition at line 31 of file [DCPEventGenerator.hpp](#).

24.16.2.2 friend struct DCPParserHelper::doEndDCP [friend]

Definition at line 32 of file [DCPEventGenerator.hpp](#).

24.16.2.3 friend class DCPParser [friend]

Definition at line 33 of file [DCPEventGenerator.hpp](#).

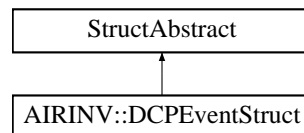
The documentation for this class was generated from the following files:

- [airinv/command/vault/DCPEventGenerator.hpp](#)
- [airinv/command/vault/DCPEventGenerator.cpp](#)

24.17 AIRINV::DCPEventStruct Struct Reference

```
#include <airinv/bom/DCPEventStruct.hpp>
```

Inheritance diagram for AIRINV::DCPEventStruct:



Public Member Functions

- [DCPEventStruct](#) ()
- stdair::Date_T [getDate](#) () const
- stdair::Duration_T [getTime](#) () const
- const std::string [describe](#) () const
- const unsigned int [getAirlineListSize](#) () const
- const unsigned int [getClassCodeListSize](#) () const
- const stdair::AirlineCode_T & [getFirstAirlineCode](#) () const
- void [beginAirline](#) ()
- bool [hasNotReachedEndAirline](#) () const
- stdair::AirlineCode_T [getCurrentAirlineCode](#) () const
- void [iterateAirline](#) ()
- const std::string & [getFirstClassCode](#) () const
- void [beginClassCode](#) ()
- bool [hasNotReachedEndClassCode](#) () const
- std::string [getCurrentClassCode](#) () const
- void [iterateClassCode](#) ()

Public Attributes

- stdair::year_t [_itYear](#)
- stdair::month_t [_itMonth](#)
- stdair::day_t [_itDay](#)
- stdair::hour_t [_itHours](#)
- stdair::minute_t [_itMinutes](#)
- stdair::second_t [_itSeconds](#)
- stdair::AirlineCodeList_T::iterator [_itCurrentAirlineCode](#)
- stdair::ClassList_StringList_T::iterator [_itCurrentClassCode](#)
- stdair::AirportCode_T [_origin](#)
- stdair::AirportCode_T [_destination](#)
- stdair::Date_T [_dateRangeStart](#)
- stdair::Date_T [_dateRangeEnd](#)
- stdair::Duration_T [_timeRangeStart](#)
- stdair::Duration_T [_timeRangeEnd](#)
- stdair::CabinCode_T [_cabinCode](#)
- stdair::CityCode_T [_pos](#)
- stdair::ChannelLabel_T [_channel](#)
- stdair::DayDuration_T [_advancePurchase](#)
- stdair::SaturdayStay_T [_saturdayStay](#)
- stdair::ChangeFees_T [_changeFees](#)
- stdair::NonRefundable_T [_nonRefundable](#)

- stdair::DayDuration_T [_minimumStay](#)
- stdair::PriceValue_T [_DCP](#)
- stdair::AirlineCode_T [_airlineCode](#)
- stdair::ClassCode_T [_classCode](#)
- stdair::AirlineCodeList_T [_airlineCodeList](#)
- stdair::ClassList_StringList_T [_classCodeList](#)

24.17.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

Definition at line 21 of file [DCPEventStruct.hpp](#).

24.17.2 Constructor & Destructor Documentation

24.17.2.1 AIRINV::DCPEventStruct::DCPEventStruct ()

Default constructor.

Definition at line 18 of file [DCPEventStruct.cpp](#).

24.17.3 Member Function Documentation

24.17.3.1 stdair::Date_T AIRINV::DCPEventStruct::getDate () const

Get the date from the staging details.

Definition at line 38 of file [DCPEventStruct.cpp](#).

References [_itDay](#), [_itMonth](#), and [_itYear](#).

24.17.3.2 stdair::Duration_T AIRINV::DCPEventStruct::getTime () const

Get the time from the staging details.

Definition at line 44 of file [DCPEventStruct.cpp](#).

References [_itHours](#), [_itMinutes](#), and [_itSeconds](#).

24.17.3.3 const std::string AIRINV::DCPEventStruct::describe () const

Display of the structure.

Definition at line 53 of file [DCPEventStruct.cpp](#).

References [_advancePurchase](#), [_airlineCodeList](#), [_cabinCode](#), [_changeFees](#), [_channel](#), [_classCodeList](#), [_dateRangeEnd](#), [_dateRangeStart](#), [_DCP](#), [_destination](#), [_minimumStay](#), [_nonRefundable](#), [_origin](#), [_pos](#), [_saturdayStay](#), [_timeRangeEnd](#), and [_timeRangeStart](#).

24.17.3.4 const unsigned int AIRINV::DCPEventStruct::getAirlineListSize () const [inline]

Get the size of the airline code list.

Definition at line 37 of file [DCPEventStruct.hpp](#).

References [_airlineCodeList](#).

24.17.3.5 const unsigned int AIRINV::DCPEventStruct::getClassCodeListSize () const [inline]

Get the size of the class code list.

Definition at line 42 of file [DCPEventStruct.hpp](#).

References [_classCodeList](#).

24.17.3.6 const stdair::AirlineCode_T & AIRINV::DCPEventStruct::getFirstAirlineCode () const

Get the first airline code.

Definition at line 87 of file [DCPEventStruct.cpp](#).

References [_airlineCodeList](#).

24.17.3.7 void AIRINV::DCPEventStruct::beginAirline ()

Initialise the internal iterators on airline code: The current iterator is set on the first airline code, the next iterator is set on the second one.

Definition at line 95 of file [DCPEventStruct.cpp](#).

References [_airlineCodeList](#), and [_itCurrentAirlineCode](#).

24.17.3.8 bool AIRINV::DCPEventStruct::hasNotReachedEndAirline () const

States whether or not the end of the (airline code) list has been reached.

Definition at line 100 of file [DCPEventStruct.cpp](#).

References [_airlineCodeList](#), and [_itCurrentAirlineCode](#).

24.17.3.9 stdair::AirlineCode_T AIRINV::DCPEventStruct::getCurrentAirlineCode () const

Get the current element (airline code).

Definition at line 106 of file [DCPEventStruct.cpp](#).

References [_airlineCodeList](#), and [_itCurrentAirlineCode](#).

24.17.3.10 void AIRINV::DCPEventStruct::iterateAirline ()

Iterate for one element (airline code): increment both internal iterators on Buckets.

Definition at line 112 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#), and [_itCurrentAirlineCode](#).

24.17.3.11 const std::string & AIRINV::DCPEventStruct::getFirstClassCode () const

Get the first class code list as a string.

Definition at line 119 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#).

24.17.3.12 void AIRINV::DCPEventStruct::beginClassCode ()

Initialise the internal iterators on class code: The current iterator is set on the first class code, the next iterator is set on the second one.

Definition at line 127 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#), and [_itCurrentClassCode](#).

24.17.3.13 bool AIRINV::DCPEventStruct::hasNotReachedEndClassCode () const

States whether or not the end of the (class code) list has been reached.

Definition at line 132 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#), and [_itCurrentClassCode](#).

24.17.3.14 std::string AIRINV::DCPEventStruct::getCurrentClassCode () const

Get the current element (class code).

Definition at line 138 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#), and [_itCurrentClassCode](#).

24.17.3.15 void AIRINV::DCPEventStruct::iterateClassCode ()

Iterate for one element (classCode): increment both internal iterators on Buckets.

Definition at line 145 of file [DCPEventStruct.cpp](#).

References [_classCodeList](#), and [_itCurrentClassCode](#).

24.17.4 Member Data Documentation

24.17.4.1 stdair::year_t AIRINV::DCPEventStruct::_itYear

Staging Date.

Definition at line 87 of file [DCPEventStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.17.4.2 stdair::month_t AIRINV::DCPEventStruct::_itMonth

Definition at line 88 of file [DCPEventStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.17.4.3 stdair::day_t AIRINV::DCPEventStruct::_itDay

Definition at line 89 of file [DCPEventStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.17.4.4 stdair::hour_t AIRINV::DCPEventStruct::_itHours

Staging Time.

Definition at line 93 of file [DCPEventStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.17.4.5 stdair::minute_t AIRINV::DCPEventStruct::_itMinutes

Definition at line 94 of file [DCPEventStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.17.4.6 stdair::second_t AIRINV::DCPEventStruct::_itSeconds

Definition at line 95 of file [DCPEventStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.17.4.7 stdair::AirlineCodeList.T::iterator AIRINV::DCPEventStruct::_itCurrentAirlineCode

Iterator for the current airline code list.

Definition at line 98 of file [DCPEventStruct.hpp](#).

Referenced by [beginAirline\(\)](#), [getCurrentAirlineCode\(\)](#), [hasNotReachedEndAirline\(\)](#), and [iterateAirline\(\)](#).

24.17.4.8 stdair::ClassList_StringList.T::iterator AIRINV::DCPEventStruct::_itCurrentClassCode

Iterator for the current class code.

Definition at line 101 of file [DCPEventStruct.hpp](#).

Referenced by [beginClassCode\(\)](#), [getCurrentClassCode\(\)](#), [hasNotReachedEndClassCode\(\)](#), and [iterateClassCode\(\)](#).

24.17.4.9 stdair::AirportCode_T AIRINV::DCPEventStruct::_origin

Origin.

Definition at line 104 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.10 stdair::AirportCode_T AIRINV::DCPEventStruct::_destination

Destination.

Definition at line 107 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.11 stdair::Date_T AIRINV::DCPEventStruct::_dateRangeStart

Start Range date available for this DCP event.

Definition at line 110 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.12 stdair::Date_T AIRINV::DCPEventStruct::_dateRangeEnd

Start Range date available for this DCP event.

Definition at line 113 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.13 stdair::Duration_T AIRINV::DCPEventStruct::_timeRangeStart

Start time from the time range available for this DCP event.

Definition at line 116 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.14 stdair::Duration_T AIRINV::DCPEventStruct::_timeRangeEnd

End time from the time range available for this DCP event.

Definition at line 119 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.15 stdair::CabinCode_T AIRINV::DCPEventStruct::_cabinCode

Cabin code.

Definition at line 122 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.16 stdair::CityCode_T AIRINV::DCPEventStruct::_pos

Point-of-sale.

Definition at line 125 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.17 stdair::ChannelLabel_T AIRINV::DCPEventStruct::_channel

Channel distribution.

Definition at line 128 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.18 stdair::DayDuration_T AIRINV::DCPEventStruct::_advancePurchase

Number of days that the ticket is sold before the flightDate.

Definition at line 131 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.19 stdair::SaturdayStay_T AIRINV::DCPEventStruct::_saturdayStay

Boolean saying whether a saturday is considered during the stay .

Definition at line 134 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.20 stdair::ChangeFees_T AIRINV::DCPEventStruct::_changeFees

Boolean saying whether the change fees option is requested or not.

Definition at line 137 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.21 stdair::NonRefundable_T AIRINV::DCPEventStruct::_nonRefundable

Boolean saying whether the refundable option is requested or not.

Definition at line 140 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.22 stdair::DayDuration_T AIRINV::DCPEventStruct::_minimumStay

Number of days that the customer spent into the destination city.

Definition at line 143 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.23 stdair::PriceValue_T AIRINV::DCPEventStruct::_DCP

Price.

Definition at line 146 of file [DCPEventStruct.hpp](#).

Referenced by [describe\(\)](#).

24.17.4.24 stdair::AirlineCode_T AIRINV::DCPEventStruct::_airlineCode

Airline code

Definition at line 149 of file [DCPEventStruct.hpp](#).

24.17.4.25 stdair::ClassCode_T AIRINV::DCPEventStruct::_classCode

Code

Definition at line 152 of file [DCPEventStruct.hpp](#).

24.17.4.26 stdair::AirlineCodeList_T AIRINV::DCPEventStruct::_airlineCodeList

Airline Code List

Definition at line 155 of file [DCPEventStruct.hpp](#).

Referenced by [beginAirline\(\)](#), [describe\(\)](#), [getAirlineListSize\(\)](#), [getCurrentAirlineCode\(\)](#), [getFirstAirlineCode\(\)](#), and [hasNotReachedEndAirline\(\)](#).

24.17.4.27 stdair::ClassList_StringList_T AIRINV::DCPEventStruct::_classCodeList

Numbers of different Airline Codes Class Code List

Definition at line 161 of file [DCPEventStruct.hpp](#).

Referenced by [beginClassCode\(\)](#), [describe\(\)](#), [getClassCodeListSize\(\)](#), [getCurrentClassCode\(\)](#), [getFirstClassCode\(\)](#), [hasNotReachedEndClassCode\(\)](#), [iterateAirline\(\)](#), and [iterateClassCode\(\)](#).

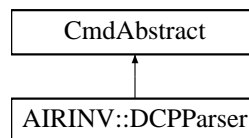
The documentation for this struct was generated from the following files:

- [airinv/bom/DCPEventStruct.hpp](#)
- [airinv/bom/DCPEventStruct.cpp](#)

24.18 AIRINV::DCPParser Class Reference

```
#include <airinv/command/vault/DCPParser.hpp>
```

Inheritance diagram for AIRINV::DCPParser:



Static Public Member Functions

- static void [DCPRuleGeneration](#) (const stdair::Filename_T &, stdair::BomRoot &)

24.18.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 19 of file [DCPParser.hpp](#).

24.18.2 Member Function Documentation

24.18.2.1 void AIRINV::DCPParser::DCPRuleGeneration (const stdair::Filename_T & iFilename, stdair::BomRoot & ioBomRoot) [static]

Parses the CSV file describing the DCPs for the simulator, and generates the event structures accordingly.

Parameters

<i>const</i>	stdair::Filename_T& The file-name of the CSV-formatted DCP input file.
<i>stdair::Bom-Root&</i>	Root of the BOM tree.

Definition at line 16 of file [DCPParser.cpp](#).

References [AIRINV::DCPRuleFileParser::generateDCPRules\(\)](#).

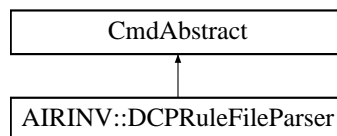
The documentation for this class was generated from the following files:

- [airinv/command/vault/DCPParser.hpp](#)
- [airinv/command/vault/DCPParser.cpp](#)

24.19 AIRINV::DCPRuleFileParser Class Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPRuleFileParser:



Public Member Functions

- [DCPRuleFileParser](#) (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool [generateDCPRules](#) ()

24.19.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 337 of file [DCPParserHelper.hpp](#).

24.19.2 Constructor & Destructor Documentation

24.19.2.1 AIRINV::DCPRuleFileParser::DCPRuleFileParser (stdair::BomRoot & ioBomRoot, const stdair::Filename_T & iFilename)

Constructor.

Definition at line 572 of file [DCPParserHelper.cpp](#).

24.19.3 Member Function Documentation

24.19.3.1 bool AIRINV::DCPRuleFileParser::generateDCPRules ()

Parse the input file and generate the Inventories.

Definition at line 593 of file [DCPParserHelper.cpp](#).

Referenced by [AIRINV::DCPParser::DCPRuleGeneration\(\)](#).

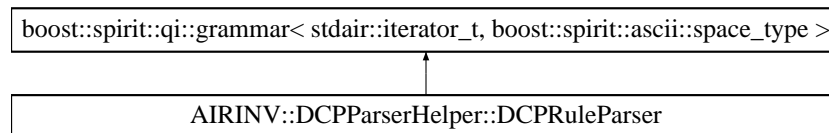
The documentation for this class was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.20 AIRINV::DCPParserHelper::DCPRuleParser Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::DCPRuleParser:



Public Member Functions

- [DCPRuleParser](#) (stdair::BomRoot &, DCPRuleStruct &)

Public Attributes

- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [start](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [comments](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [DCP_rule](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [DCP_rule_end](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [DCP_key](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [DCP_id](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [origin](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [destination](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [dateRangeStart](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [dateRangeEnd](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [date](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [timeRangeStart](#)

- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [timeRangeEnd](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [time](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [position](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [cabinCode](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [channel](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [advancePurchase](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [saturdayStay](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [changeFees](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [nonRefundable](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [minimumStay](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [DCP](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [segment](#)
- boost::spirit::qi::rule
< stdair::iterator_t,
boost::spirit::ascii::space_type > [list_class](#)
- stdair::BomRoot & [_bomRoot](#)
- DCPRuleStruct & [_DCPRule](#)

24.20.1 Detailed Description

DCP: DCPID; OriginCity; DestinationCity; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; AirlineCode; Class;

DCPID OriginCity (3-char airport code) DestinationCity (3-char airport code) DateRangeStart (yyyy-mm-dd) DateRangeEnd (yyyy-mm-dd) DepartureTimeRangeStart (hh:mm) DepartureTimeRangeEnd (hh:mm) POS (3-char position city) Cabin Code (1-char cabin code) Channel (D=direct, I=indirect, N=online, F=offline) AdvancePurchase SaturdayNight (T=True, F=False) ChangeFees (T=True, F=False) NonRefundable (T=True, F=False) MinimumStay Price AirlineCode (2-char airline code) ClassList (List of 1-char class code)

Grammar: Demand ::= PrefDepDate ',' Origin ',' Destination ',' PassengerType ',' DemandParams ',' PosDist ',' ChannelDist ',' TripDist ',' StayDist ',' FfDist ',' PrefDepTimeDist ',' minWTP ',' TimeValueDist ',' DtdDist EndOfDemand PrefDepDate ::= date PassengerType ::= 'T' | 'F' DemandParams ::= DemandMean ',' DemandStdDev PosDist ::= PosPair (' PosPair)* PosPair ::= PosCode ':' PosShare PosCode ::= AirportCode | "row" PosShare ::=

```

real ChannelDist ::= ChannelPair (',' ChannelPair)* ChannelPair ::= Channel_Code ':' ChannelShare ChannelCode
::= "DF" | "DN" | "IF" | "IN" ChannelShare ::= real TripDist ::= TripPair (',' TripPair)* TripPair ::= TripCode ':' TripShare
TripCode ::= "RO" | "RI" | "OW" TripShare ::= real StayDist ::= StayPair (',' StayPair)* StayPair ::= [0;3]-digit-integer
':' stay_share StayShare ::= real FFDist ::= FF_Pair (',' FF_Pair)* FF_Pair ::= FFCODE ':' FFShare FFCODE ::= 'P' | 'G'
| 'S' | 'M' | 'N' FFShare ::= real PrefDepTimeDist ::= PrefDepTimePair (',' PrefDepTimePair)* PrefDepTimePair ::=
time ':' PrefDepTimeShare PrefDepTimeShare ::= real minWTP ::= real TimeValueDist ::= TimeValuePair (',' Time-
ValuePair)* TimeValuePair ::= [0;2]-digit-integer ':' TimeValueShare TimeValueShare ::= real DTDDist ::= DTDPair
(',' DTDPair)* DTDPair ::= real ':' DTDSHare DTDSHare ::= real EndOfDemand ::= ';' Grammar for the DCP-Rule
parser.

```

Definition at line 304 of file [DCPParserHelper.hpp](#).

24.20.2 Constructor & Destructor Documentation

24.20.2.1 AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser (stdair::BomRoot & ioBomRoot, DCPRuleStruct & ioDCPRule)

Definition at line 453 of file [DCPParserHelper.cpp](#).

References [_bomRoot](#), [_DCPRule](#), [advancePurchase](#), [cabinCode](#), [changeFees](#), [channel](#), [comments](#), [date](#), [dateRangeEnd](#), [dateRangeStart](#), [AIRINV::DCPParserHelper::day_p](#), [DCP](#), [DCP_id](#), [DCP_key](#), [DCP_rule](#), [DCP_rule_end](#), [destination](#), [AIRINV::DCPParserHelper::hour_p](#), [list_class](#), [minimumStay](#), [AIRINV::DCPParserHelper::minute_p](#), [AIRINV::DCPParserHelper::month_p](#), [nonRefundable](#), [origin](#), [position](#), [saturdayStay](#), [AIRINV::DCPParserHelper::second_p](#), [segment](#), [start](#), [time](#), [timeRangeEnd](#), [timeRangeStart](#), [AIRINV::DCPParserHelper::uint1_4_p](#), and [AIRINV::DCPParserHelper::year_p](#).

24.20.3 Member Data Documentation

24.20.3.1 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::start

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.2 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::comments

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.3 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_rule

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.4 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_rule_end

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.5 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_key

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.6 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_id

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.7 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::origin

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.8 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::destination

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.9 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::dateRangeStart

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.10 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::dateRangeEnd

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.11 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::date

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.12 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::timeRangeStart

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.13 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::timeRangeEnd

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.14 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::time

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.15 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::position`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.16 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::cabinCode`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.17 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::channel`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.18 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::advancePurchase`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.19 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::saturdayStay`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.20 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::changeFees`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.21 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::nonRefundable`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.22 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::minimumStay`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.23 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.24 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCP-
RuleParser::segment`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.25 `boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCP-
RuleParser::list_class`

Definition at line 313 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.26 `stdair::BomRoot& AIRINV::DCPParserHelper::DCPRuleParser::_bomRoot`

Definition at line 320 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

24.20.3.27 `DCPRuleStruct& AIRINV::DCPParserHelper::DCPRuleParser::_DCPRule`

Definition at line 321 of file [DCPParserHelper.hpp](#).

Referenced by [DCPRuleParser\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.21 AIRINV::DefaultMap Struct Reference

```
#include <airinv/basic/BasConst_Curves.hpp>
```

Static Public Member Functions

- static [FRAT5Curve_T createPickupFRAT5Curve\(\)](#)

24.21.1 Detailed Description

Default PoS probability mass.

Definition at line 16 of file [BasConst_Curves.hpp](#).

24.21.2 Member Function Documentation

24.21.2.1 `FRAT5Curve_T AIRINV::DefaultMap::createPickupFRAT5Curve() [static]`

Definition at line 16 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

- [airinv/basic/BasConst_Curves.hpp](#)
- [airinv/basic/BasConst.cpp](#)

24.22 AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT > Struct Template Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Public Member Functions

- [definition](#) ([InventoryParser](#) const &self)
- [boost::spirit::classic::rule](#)
< ScannerT > const & [start](#) () const

Public Attributes

- [boost::spirit::classic::rule](#)
< ScannerT > [flight_date_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [not_to_be_parsed](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_date](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_date_end](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_key](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [airline_code](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_number](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_type_code](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_visibility_code](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [date](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_key](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_cabin_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [bucket_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [bucket_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [time](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [segment_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [segment](#)

- boost::spirit::classic::rule
 < ScannerT > [segment_key](#)
- boost::spirit::classic::rule
 < ScannerT > [full_segment_cabin_details](#)
- boost::spirit::classic::rule
 < ScannerT > [segment_cabin_list](#)
- boost::spirit::classic::rule
 < ScannerT > [segment_cabin_key](#)
- boost::spirit::classic::rule
 < ScannerT > [segment_cabin_details](#)
- boost::spirit::classic::rule
 < ScannerT > [class_list](#)
- boost::spirit::classic::rule
 < ScannerT > [class_key](#)
- boost::spirit::classic::rule
 < ScannerT > [parent_subclass_code](#)
- boost::spirit::classic::rule
 < ScannerT > [class_protection](#)
- boost::spirit::classic::rule
 < ScannerT > [class_nego](#)
- boost::spirit::classic::rule
 < ScannerT > [class_details](#)
- boost::spirit::classic::rule
 < ScannerT > [family_cabin_list](#)
- boost::spirit::classic::rule
 < ScannerT > [family_cabin_details](#)

24.22.1 Detailed Description

template<typename ScannerT>struct AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >

Definition at line 460 of file [InventoryParserHelper.hpp](#).

24.22.2 Constructor & Destructor Documentation

24.22.2.1 template<typename ScannerT > AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition (InventoryParser const & self)

Definition at line 872 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::airline_code_p\(\)](#), [AIRINV::InventoryParserHelper::airport_p\(\)](#), [AIRINV::InventoryParserHelper::cabin_code_p\(\)](#), [AIRINV::InventoryParserHelper::class_code_list_p\(\)](#), [AIRINV::InventoryParserHelper::class_code_p\(\)](#), [AIRINV::InventoryParserHelper::day_p\(\)](#), [AIRINV::InventoryParserHelper::family_code_p\(\)](#), [AIRINV::InventoryParserHelper::flight_number_p\(\)](#), [AIRINV::InventoryParserHelper::hours_p\(\)](#), [AIRINV::InventoryParserHelper::minutes_p\(\)](#), [AIRINV::InventoryParserHelper::month_p\(\)](#), [AIRINV::InventoryParserHelper::seconds_p\(\)](#), [AIRINV::InventoryParserHelper::uint1_2_p\(\)](#), [AIRINV::InventoryParserHelper::uint1_3_p\(\)](#), and [AIRINV::InventoryParserHelper::year_p\(\)](#).

24.22.3 Member Function Documentation

24.22.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::start () const

Entry point of the parser.

Definition at line 1078 of file [InventoryParserHelper.cpp](#).

24.22.4 Member Data Documentation

24.22.4.1 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.2 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::not_to_be_parsed`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.3 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.4 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date_end`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.5 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_key`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.6 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::airline_code`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.7 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_number`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.8 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_type_code`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.9 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_visibility_code`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.10 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::date`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.11 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.12 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.13 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_key`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.14 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.15 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_cabin_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.16 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_cabin_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.17 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::bucket_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.18 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::bucket_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.19 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::time`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.20 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.21 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.22 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_key`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.23 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::full_segment_cabin_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.24 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_cabin_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.25 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_cabin_key`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.26 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_cabin_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.27 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::class_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.28 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::class_key`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.29 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::parent_subclass_code`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.30 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::class_protection`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.31 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::class_nego`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.32 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::class_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.33 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::family_cabin_list`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

24.22.4.34 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::family_cabin_details`

Definition at line 464 of file [InventoryParserHelper.hpp](#).

The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.23 AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Public Member Functions

- [definition](#) ([FlightPeriodParser](#) const &self)
- [boost::spirit::classic::rule](#)
< ScannerT > const & [start](#) () const

Public Attributes

- [boost::spirit::classic::rule](#)
< ScannerT > [flight_period_list](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [not_to_be_parsed](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_period](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_period_end](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_key](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [airline_code](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [flight_number](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [date](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [dow](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [time](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [date_offset](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_key](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [leg_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [segment_section](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [segment_key](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [full_segment_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [segment_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [full_family_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [family_cabin_details](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [generic_segment](#)
- [boost::spirit::classic::rule](#)
< ScannerT > [specific_segment_list](#)

24.23.1 Detailed Description

template<typename ScannerT>struct AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >

Definition at line 255 of file [ScheduleParserHelper.hpp](#).

24.23.2 Constructor & Destructor Documentation

24.23.2.1 template<typename ScannerT > AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition (FlightPeriodParser const & self)

Definition at line 475 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::ScheduleParserHelper::airline_code_p\(\)](#), [AIRINV::ScheduleParserHelper::airport_p\(\)](#), [AIRINV::ScheduleParserHelper::cabin_code_p\(\)](#), [AIRINV::ScheduleParserHelper::class_code_list_p\(\)](#), [AIRINV::ScheduleParserHelper::day_p\(\)](#), [AIRINV::ScheduleParserHelper::dow_p\(\)](#), [AIRINV::ScheduleParserHelper::family_code_p\(\)](#), [AIRINV::ScheduleParserHelper::flight_number_p\(\)](#), [AIRINV::ScheduleParserHelper::hours_p\(\)](#), [AIRINV::ScheduleParserHelper::int1_p\(\)](#), [AIRINV::ScheduleParserHelper::minutes_p\(\)](#), [AIRINV::ScheduleParserHelper::month_p\(\)](#), [AIRINV::ScheduleParserHelper::seconds_p\(\)](#), and [AIRINV::ScheduleParserHelper::year_p\(\)](#).

24.23.3 Member Function Documentation

24.23.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::start () const

Entry point of the parser.

Definition at line 617 of file [ScheduleParserHelper.cpp](#).

24.23.4 Member Data Documentation

24.23.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period_list

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.2 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::not_to_be_parsed

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.3 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.4 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period_end

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.5 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_key

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.6 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::airline_code`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.7 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_number`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.8 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.9 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::dow`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.10 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::time`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.11 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date_offset`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.12 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.13 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg_key`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.14 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.15 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg_cabin_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.16 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::segment_section`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.17 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::segment_key`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.18 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::full_segment_cabin_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.19 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::segment_cabin_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.20 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::full_family_cabin_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.21 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::family_cabin_details`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.22 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::generic_segment`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

24.23.4.23 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::specific_segment_list`

Definition at line 259 of file [ScheduleParserHelper.hpp](#).

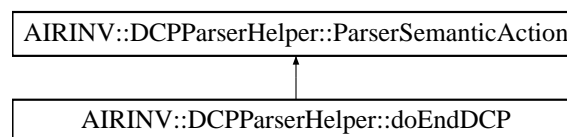
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.24 AIRINV::DCPParserHelper::doEndDCP Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::doEndDCP:



Public Member Functions

- [doEndDCP](#) (stdair::BomRoot &, DCPRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- stdair::BomRoot & [_bomRoot](#)
- DCPRuleStruct & [_DCPRule](#)

24.24.1 Detailed Description

Mark the end of the DCP-rule parsing.

Definition at line 218 of file [DCPParserHelper.hpp](#).

24.24.2 Constructor & Destructor Documentation

24.24.2.1 AIRINV::DCPParserHelper::doEndDCP::doEndDCP (stdair::BomRoot & ioBomRoot, DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 399 of file [DCPParserHelper.cpp](#).

24.24.3 Member Function Documentation

24.24.3.1 void AIRINV::DCPParserHelper::doEndDCP::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 406 of file [DCPParserHelper.cpp](#).

References [_bomRoot](#), and [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.24.4 Member Data Documentation

24.24.4.1 stdair::BomRoot& AIRINV::DCPParserHelper::doEndDCP::_bomRoot

Actor Specific Context.

Definition at line 226 of file [DCPParserHelper.hpp](#).

Referenced by [operator\(\)\(\)](#).

24.24.4.2 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)\(\)](#), and [operator\(\)\(\)](#).

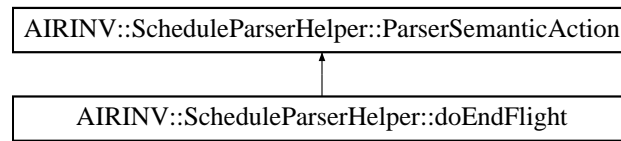
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.25 AIRINV::ScheduleParserHelper::doEndFlight Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::doEndFlight:



Public Member Functions

- [doEndFlight](#) (stdair::BomRoot & [FlightPeriodStruct](#) &)
- void [operator\(\)](#) (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & [_bomRoot](#)
- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.25.1 Detailed Description

Mark the end of the flight-period parsing.

Definition at line 192 of file [ScheduleParserHelper.hpp](#).

24.25.2 Constructor & Destructor Documentation

24.25.2.1 AIRINV::ScheduleParserHelper::doEndFlight::doEndFlight (stdair::BomRoot & *ioBomRoot*, [FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 376 of file [ScheduleParserHelper.cpp](#).

24.25.3 Member Function Documentation

24.25.3.1 void AIRINV::ScheduleParserHelper::doEndFlight::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 384 of file [ScheduleParserHelper.cpp](#).

References [_bomRoot](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_legAlreadyDefined](#), [AIRINV::FlightPeriodStruct::_legList](#), and [AIRINV::FlightPeriodStruct::describe\(\)](#).

24.25.4 Member Data Documentation

24.25.4.1 stdair::BomRoot& AIRINV::ScheduleParserHelper::doEndFlight::_bomRoot

Actor Specific Context.

Definition at line 198 of file [ScheduleParserHelper.hpp](#).

Referenced by [operator\(\)\(\)](#).

24.25.4.2 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [operator\(\)](#).

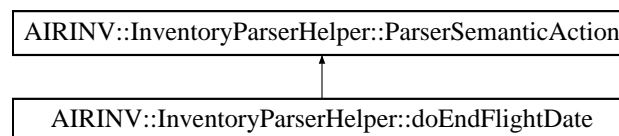
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.26 AIRINV::InventoryParserHelper::doEndFlightDate Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::doEndFlightDate:



Public Member Functions

- [doEndFlightDate](#) (stdair::BomRoot &, [FlightDateStruct](#) &, unsigned int &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- stdair::BomRoot & [_bomRoot](#)
- unsigned int & [_nbOfFlights](#)
- [FlightDateStruct](#) & [_flightDate](#)

24.26.1 Detailed Description

Mark the end of the inventory parsing.

Definition at line 425 of file [InventoryParserHelper.hpp](#).

24.26.2 Constructor & Destructor Documentation

24.26.2.1 AIRINV::InventoryParserHelper::doEndFlightDate::doEndFlightDate (stdair::BomRoot & ioBomRoot, FlightDateStruct & ioFlightDate, unsigned int & ioNbOfFlights)

Actor Constructor.

Definition at line 746 of file [InventoryParserHelper.cpp](#).

24.26.3 Member Function Documentation

24.26.3.1 void AIRINV::InventoryParserHelper::doEndFlightDate::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 755 of file [InventoryParserHelper.cpp](#).

References [_bomRoot](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itSegment](#), [_nbOfFlights](#), and [AIRINV::FlightDateStruct::_segmentList](#).

24.26.4 Member Data Documentation

24.26.4.1 stdair::BomRoot& AIRINV::InventoryParserHelper::doEndFlightDate::_bomRoot

Actor Specific Context.

Definition at line 432 of file [InventoryParserHelper.hpp](#).

Referenced by [operator\(\)\(\)](#).

24.26.4.2 unsigned int& AIRINV::InventoryParserHelper::doEndFlightDate::_nbOfFlights

Definition at line 433 of file [InventoryParserHelper.hpp](#).

Referenced by [operator\(\)\(\)](#).

24.26.4.3 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailablity::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::](#)

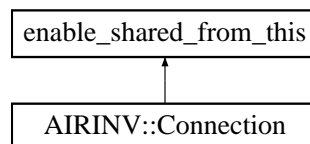
`::storeParentClassCode::operator()()`, `AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()()`, `AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()()`, `AIRINV::InventoryParserHelper::storeProtection::operator()()`, `AIRINV::InventoryParserHelper::storeNego::operator()()`, `AIRINV::InventoryParserHelper::storeNoShow::operator()()`, `AIRINV::InventoryParserHelper::storeOverbooking::operator()()`, `AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()()`, `AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()()`, `AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()()`, `AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()()`, `AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()()`, `AIRINV::InventoryParserHelper::storeClassETB::operator()()`, `AIRINV::InventoryParserHelper::storeClassAvailability::operator()()`, `AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()()`, `AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()()`, `AIRINV::InventoryParserHelper::storeFamilyCode::operator()()`, `AIRINV::InventoryParserHelper::storeFCClasses::operator()()`, and `operator()()`.

The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.27 enable_shared_from_this Class Reference

Inheritance diagram for `enable_shared_from_this`:



The documentation for this class was generated from the following file:

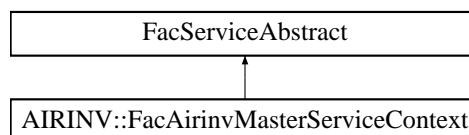
- [airinv/server/Connection.hpp](#)

24.28 AIRINV::FacAirinvMasterServiceContext Class Reference

Factory for Bucket.

```
#include <airinv/factory/FacAirinvMasterServiceContext.hpp>
```

Inheritance diagram for `AIRINV::FacAirinvMasterServiceContext`:



Public Member Functions

- `~FacAirinvMasterServiceContext ()`
- `AIRINV_Master_ServiceContext & create ()`

Static Public Member Functions

- `static`
`FacAirinvMasterServiceContext & instance ()`

Protected Member Functions

- [FacAirinvMasterServiceContext\(\)](#)

24.28.1 Detailed Description

Factory for Bucket.

Definition at line 20 of file [FacAirinvMasterServiceContext.hpp](#).

24.28.2 Constructor & Destructor Documentation

24.28.2.1 AIRINV::FacAirinvMasterServiceContext::~~FacAirinvMasterServiceContext()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacAirinvMasterServiceContext::instance\(\)](#)

Definition at line 17 of file [FacAirinvMasterServiceContext.cpp](#).

24.28.2.2 AIRINV::FacAirinvMasterServiceContext::FacAirinvMasterServiceContext() [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 44 of file [FacAirinvMasterServiceContext.hpp](#).

Referenced by [instance\(\)](#).

24.28.3 Member Function Documentation

24.28.3.1 FacAirinvMasterServiceContext & AIRINV::FacAirinvMasterServiceContext::instance() [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

[FacAirinvMasterServiceContext&](#)

Definition at line 22 of file [FacAirinvMasterServiceContext.cpp](#).

References [FacAirinvMasterServiceContext\(\)](#).

24.28.3.2 AIRINV_Master_ServiceContext & AIRINV::FacAirinvMasterServiceContext::create()

Create a new [AIRINV_Master_ServiceContext](#) object.

This new object is added to the list of instantiated objects.

Returns

[AIRINV_Master_ServiceContext&](#) The newly created object.

Definition at line 34 of file [FacAirinvMasterServiceContext.cpp](#).

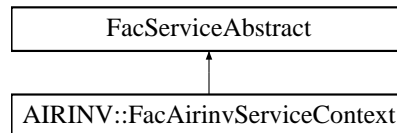
The documentation for this class was generated from the following files:

- [airinv/factory/FacAirinvMasterServiceContext.hpp](#)
- [airinv/factory/FacAirinvMasterServiceContext.cpp](#)

24.29 AIRINV::FacAirinvServiceContext Class Reference

```
#include <airinv/factory/FacAirinvServiceContext.hpp>
```

Inheritance diagram for AIRINV::FacAirinvServiceContext:



Public Member Functions

- [~FacAirinvServiceContext\(\)](#)
- [AIRINV_ServiceContext & create\(\)](#)

Static Public Member Functions

- static [FacAirinvServiceContext & instance\(\)](#)

Protected Member Functions

- [FacAirinvServiceContext\(\)](#)

24.29.1 Detailed Description

Factory for Bucket.

Definition at line 18 of file [FacAirinvServiceContext.hpp](#).

24.29.2 Constructor & Destructor Documentation

24.29.2.1 AIRINV::FacAirinvServiceContext::~~FacAirinvServiceContext()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacAirinvServiceContext::instance\(\)](#)

Definition at line 17 of file [FacAirinvServiceContext.cpp](#).

24.29.2.2 AIRINV::FacAirinvServiceContext::FacAirinvServiceContext() [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 42 of file [FacAirinvServiceContext.hpp](#).

Referenced by [instance\(\)](#).

24.29.3 Member Function Documentation

24.29.3.1 FacAirinvServiceContext & AIRINV::FacAirinvServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

[FacAirinvServiceContext&](#)

Definition at line 22 of file [FacAirinvServiceContext.cpp](#).

References [FacAirinvServiceContext\(\)](#).

24.29.3.2 AIRINV_ServiceContext & AIRINV::FacAirinvServiceContext::create ()

Create a new [AIRINV_ServiceContext](#) object.

This new object is added to the list of instantiated objects.

Returns

[AIRINV_ServiceContext&](#) The newly created object.

Definition at line 34 of file [FacAirinvServiceContext.cpp](#).

The documentation for this class was generated from the following files:

- [airinv/factory/FacAirinvServiceContext.hpp](#)
- [airinv/factory/FacAirinvServiceContext.cpp](#)

24.30 AIRINV::FacBomAbstract Class Reference

```
#include <airinv/factory/FacBomAbstract.hpp>
```

Public Types

- typedef std::vector
< [BomAbstract *](#) > [BomPool_T](#)

Static Public Member Functions

- static std::size_t [getID](#) (const [BomAbstract *](#))
- static std::size_t [getID](#) (const [BomAbstract &](#))
- static std::string [getIDString](#) (const [BomAbstract *](#))
- static std::string [getIDString](#) (const [BomAbstract &](#))

Protected Member Functions

- [FacBomAbstract](#) ()
- [FacBomAbstract](#) (const [FacBomAbstract &](#))
- virtual [~FacBomAbstract](#) ()

Protected Attributes

- [BomPool_T _pool](#)

Friends

- class [FacSupervisor](#)

24.30.1 Detailed Description

Base class for Factory layer.

Definition at line 17 of file [FacBomAbstract.hpp](#).

24.30.2 Member Typedef Documentation

24.30.2.1 `typedef std::vector<BomAbstract*> AIRINV::FacBomAbstract::BomPool_T`

Define the list (pool) of Bom objects.

Definition at line 22 of file [FacBomAbstract.hpp](#).

24.30.3 Constructor & Destructor Documentation

24.30.3.1 `AIRINV::FacBomAbstract::FacBomAbstract () [inline, protected]`

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 41 of file [FacBomAbstract.hpp](#).

24.30.3.2 `AIRINV::FacBomAbstract::FacBomAbstract (const FacBomAbstract &) [inline, protected]`

Definition at line 42 of file [FacBomAbstract.hpp](#).

24.30.3.3 `AIRINV::FacBomAbstract::~~FacBomAbstract () [protected, virtual]`

Destructor.

Definition at line 16 of file [FacBomAbstract.cpp](#).

24.30.4 Member Function Documentation

24.30.4.1 `std::size_t AIRINV::FacBomAbstract::getID (const BomAbstract * iBomAbstract_ptr) [static]`

Return the ID corresponding to the given object pointer.

Definition at line 35 of file [FacBomAbstract.cpp](#).

Referenced by [getID\(\)](#), and [getIDString\(\)](#).

24.30.4.2 `std::size_t AIRINV::FacBomAbstract::getID (const BomAbstract & iBomAbstract) [static]`

Return the ID corresponding to the given object reference.

Definition at line 43 of file [FacBomAbstract.cpp](#).

References [getID\(\)](#).

24.30.4.3 `std::string AIRINV::FacBomAbstract::getIDString (const BomAbstract * iBomAbstract_ptr) [static]`

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 48 of file [FacBomAbstract.cpp](#).

References [getID\(\)](#).

Referenced by [getIDString\(\)](#).

24.30.4.4 `std::string AIRINV::FacBomAbstract::getIDString (const BomAbstract & iBomAbstract) [static]`

Return the ID, as a string, corresponding to the given object reference.

Definition at line 56 of file [FacBomAbstract.cpp](#).

References [getIDString\(\)](#).

24.30.5 Friends And Related Function Documentation

24.30.5.1 `friend class FacSupervisor [friend]`

Definition at line 18 of file [FacBomAbstract.hpp](#).

24.30.6 Member Data Documentation

24.30.6.1 `BomPool_T AIRINV::FacBomAbstract::_pool [protected]`

List of instantiated Business Objects

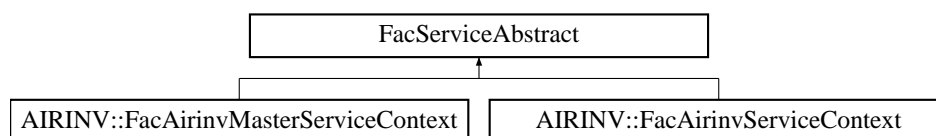
Definition at line 53 of file [FacBomAbstract.hpp](#).

The documentation for this class was generated from the following files:

- [airinv/factory/FacBomAbstract.hpp](#)
- [airinv/factory/FacBomAbstract.cpp](#)

24.31 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



The documentation for this class was generated from the following file:

- [airinv/factory/FacAirinvMasterServiceContext.hpp](#)

24.32 AIRINV::FacServiceAbstract Class Reference

```
#include <airinv/factory/FacServiceAbstract.hpp>
```

Public Types

- `typedef std::vector
< ServiceAbstract * > ServicePool_T`

Public Member Functions

- virtual [~FacServiceAbstract](#) ()
- void [clean](#) ()

Protected Member Functions

- [FacServiceAbstract](#) ()

Protected Attributes

- [ServicePool_T_pool](#)

24.32.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file [FacServiceAbstract.hpp](#).

24.32.2 Member Typedef Documentation

24.32.2.1 `typedef std::vector<ServiceAbstract*> AIRINV::FacServiceAbstract::ServicePool_T`

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

24.32.3 Constructor & Destructor Documentation

24.32.3.1 `AIRINV::FacServiceAbstract::~~FacServiceAbstract () [virtual]`

Destructor.

Definition at line 13 of file [FacServiceAbstract.cpp](#).

References [clean\(\)](#).

24.32.3.2 `AIRINV::FacServiceAbstract::FacServiceAbstract () [inline, protected]`

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file [FacServiceAbstract.hpp](#).

24.32.4 Member Function Documentation

24.32.4.1 `void AIRINV::FacServiceAbstract::clean ()`

Destroyed all the object instantiated by this factory.

Definition at line 18 of file [FacServiceAbstract.cpp](#).

References [_pool](#).

Referenced by [~FacServiceAbstract\(\)](#).

24.32.5 Member Data Documentation

24.32.5.1 ServicePool_T AIRINV::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file [FacServiceAbstract.hpp](#).

Referenced by [clean\(\)](#).

The documentation for this class was generated from the following files:

- [airinv/factory/FacServiceAbstract.hpp](#)
- [airinv/factory/FacServiceAbstract.cpp](#)

24.33 AIRINV::FacSupervisor Class Reference

```
#include <airinv/factory/FacSupervisor.hpp>
```

Public Types

- typedef std::vector
< [FacBomAbstract](#) * > [BomFactoryPool_T](#)
- typedef std::vector
< [FacServiceAbstract](#) * > [ServiceFactoryPool_T](#)

Public Member Functions

- void [registerBomFactory](#) ([FacBomAbstract](#) *)
- void [registerServiceFactory](#) ([FacServiceAbstract](#) *)
- void [cleanBomLayer](#) ()
- void [cleanServiceLayer](#) ()
- [~FacSupervisor](#) ()

Static Public Member Functions

- static [FacSupervisor](#) & [instance](#) ()
- static void [cleanFactory](#) ()

Protected Member Functions

- [FacSupervisor](#) ()
- [FacSupervisor](#) (const [FacSupervisor](#) &)

24.33.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 17 of file [FacSupervisor.hpp](#).

24.33.2 Member Typedef Documentation

24.33.2.1 typedef std::vector<[FacBomAbstract](#)*> AIRINV::FacSupervisor::BomFactoryPool_T

Define the pool (list) of factories.

Definition at line 21 of file [FacSupervisor.hpp](#).

24.33.2.2 `typedef std::vector<FacServiceAbstract*> AIRINV::FacSupervisor::ServiceFactoryPool_T`

Definition at line 22 of file [FacSupervisor.hpp](#).

24.33.3 Constructor & Destructor Documentation

24.33.3.1 `AIRINV::FacSupervisor::~~FacSupervisor ()`

Destructor

The static instance is deleted (and reset to NULL) by the static [cleanFactory\(\)](#) method.

Definition at line 41 of file [FacSupervisor.cpp](#).

References [cleanBomLayer\(\)](#), and [cleanServiceLayer\(\)](#).

24.33.3.2 `AIRINV::FacSupervisor::FacSupervisor ()` `[protected]`

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 16 of file [FacSupervisor.cpp](#).

Referenced by [instance\(\)](#).

24.33.3.3 `AIRINV::FacSupervisor::FacSupervisor (const FacSupervisor &)` `[inline, protected]`

Definition at line 66 of file [FacSupervisor.hpp](#).

24.33.4 Member Function Documentation

24.33.4.1 `FacSupervisor & AIRINV::FacSupervisor::instance ()` `[static]`

Provides the unique instance.

The singleton is instantiated when first used.

Returns

[FacSupervisor&](#)

Definition at line 20 of file [FacSupervisor.cpp](#).

References [FacSupervisor\(\)](#).

24.33.4.2 `void AIRINV::FacSupervisor::registerBomFactory (FacBomAbstract * ioFacBomAbstract_ptr)`

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#)

Parameters

<i>FacAbstract&</i>	the concrete Factory to register.
-------------------------	-----------------------------------

Definition at line 30 of file [FacSupervisor.cpp](#).

24.33.4.3 `void AIRINV::FacSupervisor::registerServiceFactory (FacServiceAbstract * ioFacServiceAbstract_ptr)`

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#).

Parameters

<i>FacService-Abstract</i> &	the concrete Factory to register.
------------------------------	-----------------------------------

Definition at line 36 of file [FacSupervisor.cpp](#).

24.33.4.4 void AIRINV::FacSupervisor::cleanBomLayer ()

Clean all created object.

Call the clean method of all the instantiated factories for the Bom layer.

Definition at line 47 of file [FacSupervisor.cpp](#).

Referenced by [cleanFactory\(\)](#), and [~FacSupervisor\(\)](#).

24.33.4.5 void AIRINV::FacSupervisor::cleanServiceLayer ()

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 61 of file [FacSupervisor.cpp](#).

Referenced by [cleanFactory\(\)](#), and [~FacSupervisor\(\)](#).

24.33.4.6 void AIRINV::FacSupervisor::cleanFactory () [static]

Clean the static instance.

The singleton is deleted.

Definition at line 75 of file [FacSupervisor.cpp](#).

References [cleanBomLayer\(\)](#), and [cleanServiceLayer\(\)](#).

The documentation for this class was generated from the following files:

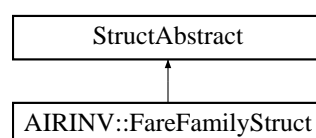
- [airinv/factory/FacSupervisor.hpp](#)
- [airinv/factory/FacSupervisor.cpp](#)

24.34 AIRINV::FareFamilyStruct Struct Reference

Utility Structure for the parsing of fare family details.

```
#include <airinv/bom/FareFamilyStruct.hpp>
```

Inheritance diagram for AIRINV::FareFamilyStruct:



Public Member Functions

- [FareFamilyStruct](#) ()
- [FareFamilyStruct](#) (const stdair::FamilyCode_T &, const stdair::ClassList_String_T &)
- void [fill](#) (stdair::FareFamily &) const
- const std::string [describe](#) () const

Public Attributes

- [stdair::FamilyCode_T _familyCode](#)
- [stdair::ClassList_String_T _classes](#)
- [BookingClassStructList_T _classList](#)

24.34.1 Detailed Description

Utility Structure for the parsing of fare family details.

Definition at line 26 of file [FareFamilyStruct.hpp](#).

24.34.2 Constructor & Destructor Documentation

24.34.2.1 AIRINV::FareFamilyStruct::FareFamilyStruct ()

Default constructor.

Definition at line 16 of file [FareFamilyStruct.cpp](#).

24.34.2.2 AIRINV::FareFamilyStruct::FareFamilyStruct (const stdair::FamilyCode_T & iFamilyCode, const stdair::ClassList_String_T & iClasses)

Main constructor.

Definition at line 23 of file [FareFamilyStruct.cpp](#).

24.34.3 Member Function Documentation

24.34.3.1 void AIRINV::FareFamilyStruct::fill (stdair::FareFamily & ioFareFamily) const

Fill the FareFamily objects with the attributes of the [FareFamilyStruct](#).

Definition at line 47 of file [FareFamilyStruct.cpp](#).

24.34.3.2 const std::string AIRINV::FareFamilyStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 29 of file [FareFamilyStruct.cpp](#).

References [_classes](#), [_classList](#), [_familyCode](#), and [AIRINV::BookingClassStruct::describe\(\)](#).

Referenced by [AIRINV::SegmentCabinStruct::describe\(\)](#).

24.34.4 Member Data Documentation

24.34.4.1 stdair::FamilyCode_T AIRINV::FareFamilyStruct::_familyCode

Definition at line 28 of file [FareFamilyStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), and [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)\(\)](#).

24.34.4.2 stdair::ClassList_String_T AIRINV::FareFamilyStruct::_classes

Definition at line 29 of file [FareFamilyStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)\(\)](#).

24.34.4.3 BookingClassStructList_T AIRINV::FareFamilyStruct::_classList

Definition at line 30 of file [FareFamilyStruct.hpp](#).

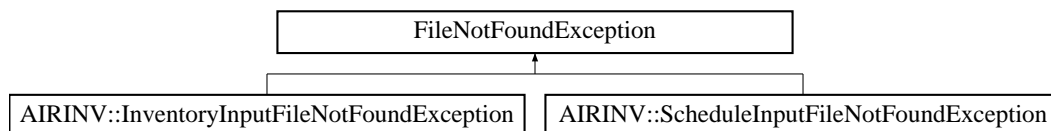
Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/bom/FareFamilyStruct.hpp](#)
- [airinv/bom/FareFamilyStruct.cpp](#)

24.35 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException:



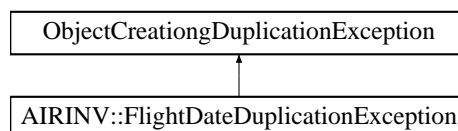
The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.36 AIRINV::FlightDateDuplicationException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::FlightDateDuplicationException:



Public Member Functions

- [FlightDateDuplicationException](#) (const std::string &iWhat)

24.36.1 Detailed Description

Duplicated flight date object.

Definition at line 90 of file [AIRINV_Types.hpp](#).

24.36.2 Constructor & Destructor Documentation

24.36.2.1 AIRINV::FlightDateDuplicationException::FlightDateDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 95 of file [AIRINV_Types.hpp](#).

The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.37 AIRINV::FlightDateHelper Class Reference

```
#include <airinv/bom/FlightDateHelper.hpp>
```

Static Public Member Functions

- static void [fillFromRouting](#) (const stdair::FlightDate &)
- static void [updateAvailabilityPool](#) (const stdair::FlightDate &, const stdair::CabinCode_T &)
- static void [updateBookingControls](#) (stdair::FlightDate &)

24.37.1 Detailed Description

Class representing the actual business functions for an airline flight-date.

Definition at line 19 of file [FlightDateHelper.hpp](#).

24.37.2 Member Function Documentation

24.37.2.1 void AIRINV::FlightDateHelper::fillFromRouting (const stdair::FlightDate & *iFlightDate*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 51 of file [FlightDateHelper.cpp](#).

24.37.2.2 void AIRINV::FlightDateHelper::updateAvailabilityPool (const stdair::FlightDate & *iFlightDate*, const stdair::CabinCode_T & *iCabinCode*) [static]

Update the availability pool of all the segment-cabins after a reservation.

Definition at line 67 of file [FlightDateHelper.cpp](#).

Referenced by [AIRINV::SegmentCabinHelper::updateFromReservation\(\)](#).

24.37.2.3 void AIRINV::FlightDateHelper::updateBookingControls (stdair::FlightDate & *ioFlightDate*) [static]

Update booking controls after optimisation.

Definition at line 22 of file [FlightDateHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::buildPseudoBidPriceVector\(\)](#), and [AIRINV::SegmentCabinHelper::updateBookingControlsUsingPseudoBidPriceVector\(\)](#).

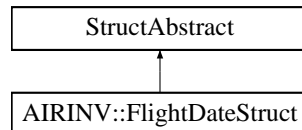
The documentation for this class was generated from the following files:

- [airinv/bom/FlightDateHelper.hpp](#)
- [airinv/bom/FlightDateHelper.cpp](#)

24.38 AIRINV::FlightDateStruct Struct Reference

```
#include <airinv/bom/FlightDateStruct.hpp>
```

Inheritance diagram for AIRINV::FlightDateStruct:



Public Member Functions

- stdair::Date_T [getDate](#) () const
- stdair::Duration_T [getTime](#) () const
- const std::string [describe](#) () const
- void [addAirport](#) (const stdair::AirportCode_T &)
- void [buildSegments](#) ()
- void [addSegmentCabin](#) (const [SegmentStruct](#) &, const [SegmentCabinStruct](#) &)
- void [addSegmentCabin](#) (const [SegmentCabinStruct](#) &)
- void [addFareFamily](#) (const [SegmentStruct](#) &, const [SegmentCabinStruct](#) &, const [FareFamilyStruct](#) &)
- void [addFareFamily](#) (const [SegmentCabinStruct](#) &, const [FareFamilyStruct](#) &)
- [FlightDateStruct](#) ()

Public Attributes

- stdair::AirlineCode_T [_airlineCode](#)
- stdair::FlightNumber_T [_flightNumber](#)
- stdair::Date_T [_flightDate](#)
- [FlightTypeCode](#) [_flightTypeCode](#)
- [FlightVisibilityCode](#) [_flightVisibilityCode](#)
- [LegStructList_T](#) [_legList](#)
- [SegmentStructList_T](#) [_segmentList](#)
- unsigned int [_itYear](#)
- unsigned int [_itMonth](#)
- unsigned int [_itDay](#)
- int [_dateOffSet](#)
- long [_itHours](#)
- long [_itMinutes](#)
- long [_itSeconds](#)
- [AirportList_T](#) [_airportList](#)
- [AirportOrderedList_T](#) [_airportOrderedList](#)
- bool [_legAlreadyDefined](#)
- [LegStruct](#) [_itLeg](#)
- [LegCabinStruct](#) [_itLegCabin](#)
- [BucketStruct](#) [_itBucket](#)
- bool [_areSegmentDefinitionsSpecific](#)
- [SegmentStruct](#) [_itSegment](#)
- [SegmentCabinStruct](#) [_itSegmentCabin](#)
- [BookingClassStruct](#) [_itBookingClass](#)

24.38.1 Detailed Description

Utility Structure for the parsing of Flight-Date structures.

Definition at line 27 of file [FlightDateStruct.hpp](#).

24.38.2 Constructor & Destructor Documentation

24.38.2.1 AIRINV::FlightDateStruct::FlightDateStruct ()

Constructor.

Definition at line 17 of file [FlightDateStruct.cpp](#).

24.38.3 Member Function Documentation

24.38.3.1 stdair::Date_T AIRINV::FlightDateStruct::getDate () const

Set the date from the staging details.

Definition at line 25 of file [FlightDateStruct.cpp](#).

References [_itDay](#), [_itMonth](#), and [_itYear](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#).

24.38.3.2 stdair::Duration_T AIRINV::FlightDateStruct::getTime () const

Set the time from the staging details.

Definition at line 30 of file [FlightDateStruct.cpp](#).

References [_itHours](#), [_itMinutes](#), and [_itSeconds](#).

Referenced by [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#).

24.38.3.3 const std::string AIRINV::FlightDateStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 37 of file [FlightDateStruct.cpp](#).

References [_airlineCode](#), [_flightDate](#), [_flightNumber](#), [_flightTypeCode](#), [_flightVisibilityCode](#), [_legList](#), [_segmentList](#), [AIRINV::SegmentStruct::describe\(\)](#), [AIRINV::LegStruct::describe\(\)](#), [AIRINV::FlightVisibilityCode::getCode\(\)](#), and [AIRINV::FlightVisibilityCode::NORMAL](#).

24.38.3.4 void AIRINV::FlightDateStruct::addAirport (const stdair::AirportCode_T & iAirport)

Add the given airport to the internal lists (if not already existing).

Definition at line 67 of file [FlightDateStruct.cpp](#).

References [_airportList](#), and [_airportOrderedList](#).

Referenced by [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#).

24.38.3.5 void AIRINV::FlightDateStruct::buildSegments ()

Build the list of [SegmentStruct](#) objects.

Definition at line 83 of file [FlightDateStruct.cpp](#).

References [_airportList](#), [_airportOrderedList](#), [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

24.38.3.6 `void AIRINV::FlightDateStruct::addSegmentCabin (const SegmentStruct & iSegment, const SegmentCabinStruct & iCabin)`

Add, to the Segment structure whose key corresponds to the given (board point, off point) pair, the specific segment cabin details (mainly, the list of the class codes).

Note that the Segment structure is retrieved from the internal list, already filled by a previous step (the [buildSegments\(\)](#) method).

Definition at line 116 of file [FlightDateStruct.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

24.38.3.7 `void AIRINV::FlightDateStruct::addSegmentCabin (const SegmentCabinStruct & iCabin)`

Add, to all the Segment structures, the general segment cabin details (mainly, the list of the class codes).

Note that the Segment structures are stored within the internal list, already filled by a previous step (the [buildSegments\(\)](#) method).

Definition at line 153 of file [FlightDateStruct.cpp](#).

References [AIRINV::SegmentStruct::_cabinList](#), and [_segmentList](#).

24.38.3.8 `void AIRINV::FlightDateStruct::addFareFamily (const SegmentStruct & iSegment, const SegmentCabinStruct & iCabin, const FareFamilyStruct & iFareFamily)`

Add, to the SegmentCabin structure whose key corresponds to the given cabin code, the specific segment fare family details (mainly, the list of the class codes).

Note that the SegmentCabin structure is retrieved from the internal list, already filled by a previous step (the [buildSegmentCabins\(\)](#) method).

Definition at line 167 of file [FlightDateStruct.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

24.38.3.9 `void AIRINV::FlightDateStruct::addFareFamily (const SegmentCabinStruct & iCabin, const FareFamilyStruct & iFareFamily)`

Add, to all the Segment structures, the general fare family sets (list of fare families).

Note that the SegmentCabin structures are stored within the internal list, already filled by a previous step (the [buildSegmentCabins\(\)](#) method).

Definition at line 231 of file [FlightDateStruct.cpp](#).

References [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), and [_segmentList](#).

24.38.4 Member Data Documentation

24.38.4.1 `stdair::AirlineCode_T AIRINV::FlightDateStruct::_airlineCode`

Definition at line 81 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)\(\)](#).

24.38.4.2 stdair::FlightNumber_T AIRINV::FlightDateStruct::_flightNumber

Definition at line 82 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#).

24.38.4.3 stdair::Date_T AIRINV::FlightDateStruct::_flightDate

Definition at line 83 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#).

24.38.4.4 FlightTypeCode AIRINV::FlightDateStruct::_flightTypeCode

Definition at line 84 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#).

24.38.4.5 FlightVisibilityCode AIRINV::FlightDateStruct::_flightVisibilityCode

Definition at line 85 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#).

24.38.4.6 LegStructList_T AIRINV::FlightDateStruct::_legList

Definition at line 86 of file [FlightDateStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

24.38.4.7 SegmentStructList_T AIRINV::FlightDateStruct::_segmentList

Definition at line 87 of file [FlightDateStruct.hpp](#).

Referenced by [addFareFamily\(\)](#), [addSegmentCabin\(\)](#), [buildSegments\(\)](#), [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

24.38.4.8 unsigned int AIRINV::FlightDateStruct::_itYear

Staging Date.

Definition at line 90 of file [FlightDateStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.38.4.9 unsigned int AIRINV::FlightDateStruct::_itMonth

Definition at line 91 of file [FlightDateStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.38.4.10 unsigned int AIRINV::FlightDateStruct::_itDay

Definition at line 92 of file [FlightDateStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.38.4.11 int AIRINV::FlightDateStruct::_dateOffSet

Definition at line 93 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#).

24.38.4.12 long AIRINV::FlightDateStruct::_itHours

Staging Time.

Definition at line 96 of file [FlightDateStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.38.4.13 long AIRINV::FlightDateStruct::_itMinutes

Definition at line 97 of file [FlightDateStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.38.4.14 long AIRINV::FlightDateStruct::_itSeconds

Definition at line 98 of file [FlightDateStruct.hpp](#).

Referenced by [getTime\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#).

24.38.4.15 AirportList_T AIRINV::FlightDateStruct::_airportList

Staging Airport List (helper to derive the list of Segment structures).

Definition at line 102 of file [FlightDateStruct.hpp](#).

Referenced by [addAirport\(\)](#), and [buildSegments\(\)](#).

24.38.4.16 AirportOrderedList_T AIRINV::FlightDateStruct::_airportOrderedList

Definition at line 103 of file [FlightDateStruct.hpp](#).

Referenced by [addAirport\(\)](#), and [buildSegments\(\)](#).

24.38.4.17 bool AIRINV::FlightDateStruct::_legAlreadyDefined

Staging Leg (resp. Cabin) structure, gathering the result of the iteration on one leg (resp. cabin).

Definition at line 107 of file [FlightDateStruct.hpp](#).

24.38.4.18 LegStruct AIRINV::FlightDateStruct::_itLeg

Definition at line 108 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

24.38.4.19 LegCabinStruct AIRINV::FlightDateStruct::_itLegCabin

Definition at line 109 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

24.38.4.20 BucketStruct AIRINV::FlightDateStruct::_itBucket

Definition at line 110 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

24.38.4.21 bool AIRINV::FlightDateStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 113 of file [FlightDateStruct.hpp](#).

24.38.4.22 SegmentStruct AIRINV::FlightDateStruct::_itSegment

Definition at line 114 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

24.38.4.23 SegmentCabinStruct AIRINV::FlightDateStruct::_itSegmentCabin

Definition at line 115 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#).

24.38.4.24 BookingClassStruct AIRINV::FlightDateStruct::_itBookingClass

Definition at line 116 of file [FlightDateStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#).

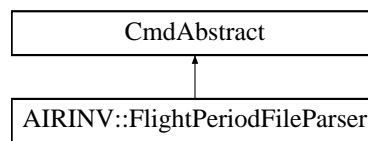
The documentation for this struct was generated from the following files:

- [airinv/bom/FlightDateStruct.hpp](#)
- [airinv/bom/FlightDateStruct.cpp](#)

24.39 AIRINV::FlightPeriodFileParser Class Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::FlightPeriodFileParser:



Public Member Functions

- [FlightPeriodFileParser](#) (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool [generateInventories](#) ()

24.39.1 Detailed Description

Short Description

Detailed Description. Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 292 of file [ScheduleParserHelper.hpp](#).

24.39.2 Constructor & Destructor Documentation

24.39.2.1 AIRINV::FlightPeriodFileParser::FlightPeriodFileParser (stdair::BomRoot & ioBomRoot, const stdair::Filename_T & iFilename)

Constructor.

Definition at line 631 of file [ScheduleParserHelper.cpp](#).

24.39.3 Member Function Documentation

24.39.3.1 bool AIRINV::FlightPeriodFileParser::generateInventories ()

Parse the input file and generate the Inventories.

Definition at line 655 of file [ScheduleParserHelper.cpp](#).

Referenced by [AIRINV::ScheduleParser::generateInventories\(\)](#).

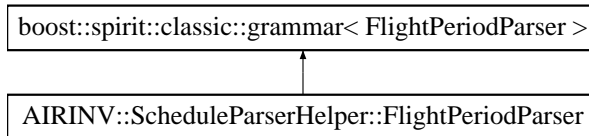
The documentation for this class was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.40 AIRINV::ScheduleParserHelper::FlightPeriodParser Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::FlightPeriodParser:



Classes

- struct [definition](#)

Public Member Functions

- [FlightPeriodParser](#) (stdair::BomRoot &, [FlightPeriodStruct](#) &)

Public Attributes

- stdair::BomRoot & [_bomRoot](#)
- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.40.1 Detailed Description

AirlineCode; FlightNumber; DateRangeStart; DateRangeEnd; DOW; (list) BoardingPoint; OffPoint; BoardingTime; DateOffset; OffTime; ElapsedTime; (list) CabinCode; Capacity; SegmentSpecificity (0 or 1); (list) (optional BoardingPoint; OffPoint); CabinCode; Classes BA; 9; 2007-04-20; 2007-04-30; 0000011; LHR; BKK; 22:00; +1; 15:15; 11:15; C; 12; M; 300; BKK; SYD; 18:10; +1; 06:05; 08:55; C; 20; M; 250; 0; C; CDIU; 1; CD; 2; IU; M; YHBKLMNOPQRSTUVWXYZ; 3; YHBKLMNOPQRSTUVWXYZ BA; 9; 2007-04-20; 2007-04-30; 1111100; LHR; SIN; 22:00; +1; 15:15; 11:15; C; 15; M; 310; SIN; SYD; 18:10; +1; 06:05; 08:55; C; 25; M; 260; 1; LHR; SIN; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ SIN; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ LHR; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ

Grammar: DOW ::= int FlightKey ::= AirlineCode ';' FlightNumber ';' DateRangeStart ';' DateRangeEnd ';' DOW
 LegKey ::= BoardingPoint ';' OffPoint LegDetails ::= BoardingTime ['/' BoardingDateOffset] ';' OffTime ['/' BoardingDateOffset] ';' Elapsed LegCabinDetails ::= CabinCode ';' Capacity Leg ::= LegKey ';' LegDetails (';' CabinDetails)+
 SegmentKey ::= BoardingPoint ';' OffPoint SegmentCabinDetails ::= CabinCode ';' Classes (';' FamilyCabinDetails)+
 FamilyCabinDetails ::= FamilyCode ';' Classes FullSegmentCabinDetails ::= (';' SegmentCabinDetails)+ GenericSegment ::= '0' (';' SegmentCabinDetails)+ SpecificSegments ::= '1' (';' SegmentKey ';' FullSegmentCabinDetails)+
 SegmentSection ::= GenericSegment | SpecificSegments FlightPeriod ::= FlightKey (';' Leg)+ ';' SegmentSection ';' EndOfFlight EndOfFlight ::= ';' Grammar for the Flight-Period parser.

Definition at line 249 of file [ScheduleParserHelper.hpp](#).

24.40.2 Constructor & Destructor Documentation

24.40.2.1 AIRINV::ScheduleParserHelper::FlightPeriodParser::FlightPeriodParser (stdair::BomRoot & ioBomRoot, FlightPeriodStruct & ioFlightPeriod)

Definition at line 466 of file [ScheduleParserHelper.cpp](#).

24.40.3 Member Data Documentation

24.40.3.1 stdair::BomRoot& AIRINV::ScheduleParserHelper::FlightPeriodParser::_bomRoot

Definition at line 273 of file [ScheduleParserHelper.hpp](#).

24.40.3.2 FlightPeriodStruct& AIRINV::ScheduleParserHelper::FlightPeriodParser::_flightPeriod

Definition at line 274 of file [ScheduleParserHelper.hpp](#).

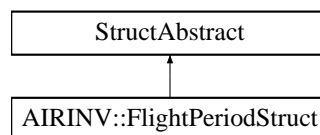
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.41 AIRINV::FlightPeriodStruct Struct Reference

```
#include <airinv/bom/FlightPeriodStruct.hpp>
```

Inheritance diagram for AIRINV::FlightPeriodStruct:



Public Member Functions

- stdair::Date_T [getDate](#) () const
- stdair::Duration_T [getTime](#) () const
- const std::string [describe](#) () const
- void [addAirport](#) (const stdair::AirportCode_T &)
- void [buildSegments](#) ()
- void [addSegmentCabin](#) (const [SegmentStruct](#) &, const [SegmentCabinStruct](#) &)
- void [addSegmentCabin](#) (const [SegmentCabinStruct](#) &)
- void [addFareFamily](#) (const [SegmentStruct](#) &, const [SegmentCabinStruct](#) &, const [FareFamilyStruct](#) &)
- void [addFareFamily](#) (const [SegmentCabinStruct](#) &, const [FareFamilyStruct](#) &)
- [FlightPeriodStruct](#) ()

Public Attributes

- stdair::AirlineCode_T [_airlineCode](#)
- stdair::FlightNumber_T [_flightNumber](#)
- stdair::DatePeriod_T [_dateRange](#)
- stdair::DoWStruct [_dow](#)
- [LegStructList_T](#) [_legList](#)
- [SegmentStructList_T](#) [_segmentList](#)
- bool [_legAlreadyDefined](#)
- [LegStruct](#) [_itLeg](#)
- [LegCabinStruct](#) [_itLegCabin](#)
- stdair::Date_T [_dateRangeStart](#)
- stdair::Date_T [_dateRangeEnd](#)
- unsigned int [_itYear](#)
- unsigned int [_itMonth](#)
- unsigned int [_itDay](#)
- int [_dateOffset](#)
- long [_itHours](#)
- long [_itMinutes](#)
- long [_itSeconds](#)

- [AirportList_T _airportList](#)
- [AirportOrderedList_T _airportOrderedList](#)
- [bool _areSegmentDefinitionsSpecific](#)
- [SegmentStruct _itSegment](#)
- [SegmentCabinStruct _itSegmentCabin](#)

24.41.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

Definition at line 24 of file [FlightPeriodStruct.hpp](#).

24.41.2 Constructor & Destructor Documentation

24.41.2.1 AIRINV::FlightPeriodStruct::FlightPeriodStruct ()

Constructor.

Definition at line 17 of file [FlightPeriodStruct.cpp](#).

24.41.3 Member Function Documentation

24.41.3.1 stdair::Date_T AIRINV::FlightPeriodStruct::getDate () const

Set the date from the staging details.

Definition at line 24 of file [FlightPeriodStruct.cpp](#).

References [_itDay](#), [_itMonth](#), and [_itYear](#).

Referenced by [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

24.41.3.2 stdair::Duration_T AIRINV::FlightPeriodStruct::getTime () const

Set the time from the staging details.

Definition at line 29 of file [FlightPeriodStruct.cpp](#).

References [_itHours](#), [_itMinutes](#), and [_itSeconds](#).

Referenced by [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

24.41.3.3 const std::string AIRINV::FlightPeriodStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 36 of file [FlightPeriodStruct.cpp](#).

References [_airlineCode](#), [_dateRange](#), [_dow](#), [_flightNumber](#), [_legList](#), [_segmentList](#), [AIRINV::SegmentStruct::describe\(\)](#), and [AIRINV::LegStruct::describe\(\)](#).

Referenced by [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

24.41.3.4 void AIRINV::FlightPeriodStruct::addAirport (const stdair::AirportCode_T & iAirport)

Add the given airport to the internal lists (if not already existing).

Definition at line 62 of file [FlightPeriodStruct.cpp](#).

References [_airportList](#), and [_airportOrderedList](#).

Referenced by [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), and [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#).

24.41.3.5 void AIRINV::FlightPeriodStruct::buildSegments ()

Build the list of [SegmentStruct](#) objects.

Definition at line 78 of file [FlightPeriodStruct.cpp](#).

References [_airportList](#), [_airportOrderedList](#), [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#).

24.41.3.6 void AIRINV::FlightPeriodStruct::addSegmentCabin (const SegmentStruct & iSegment, const SegmentCabinStruct & iCabin)

Add, to the Segment structure whose key corresponds to the given (board point, off point) pair, the specific segment cabin details (mainly, the list of the class codes).

Note that the Segment structure is retrieved from the internal list, already filled by a previous step (the [buildSegments\(\)](#) method).

Definition at line 111 of file [FlightPeriodStruct.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#).

24.41.3.7 void AIRINV::FlightPeriodStruct::addSegmentCabin (const SegmentCabinStruct & iCabin)

Add, to all the Segment structures, the general segment cabin details (mainly, the list of the class codes).

Note that the Segment structures are stored within the internal list, already filled by a previous step (the [buildSegments\(\)](#) method).

Definition at line 148 of file [FlightPeriodStruct.cpp](#).

References [AIRINV::SegmentStruct::_cabinList](#), and [_segmentList](#).

24.41.3.8 void AIRINV::FlightPeriodStruct::addFareFamily (const SegmentStruct & iSegment, const SegmentCabinStruct & iCabin, const FareFamilyStruct & iFareFamily)

Add, to the SegmentCabin structure whose key corresponds to the given cabin code, the specific segment fare family details (mainly, the list of the class codes).

Note that the SegmentCabin structure is retrieved from the internal list, already filled by a previous step (the [buildSegmentCabins\(\)](#) method).

Definition at line 161 of file [FlightPeriodStruct.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#).

24.41.3.9 void AIRINV::FlightPeriodStruct::addFareFamily (const SegmentCabinStruct & iCabin, const FareFamilyStruct & iFareFamily)

Add, to all the Segment structures, the general fare family sets (list of fare families).

Note that the SegmentCabin structures are stored within the internal list, already filled by a previous step (the [buildSegmentCabins\(\)](#) method).

Definition at line 225 of file [FlightPeriodStruct.cpp](#).

References [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::Segment-](#)

[CabinStruct::_fareFamilies](#), and [_segmentList](#).

24.41.4 Member Data Documentation

24.41.4.1 stdair::AirlineCode_T AIRINV::FlightPeriodStruct::_airlineCode

Definition at line 80 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

24.41.4.2 stdair::FlightNumber_T AIRINV::FlightPeriodStruct::_flightNumber

Definition at line 81 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#).

24.41.4.3 stdair::DatePeriod_T AIRINV::FlightPeriodStruct::_dateRange

Definition at line 82 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

24.41.4.4 stdair::DoWStruct AIRINV::FlightPeriodStruct::_dow

Definition at line 83 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#).

24.41.4.5 LegStructList_T AIRINV::FlightPeriodStruct::_legList

Definition at line 84 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

24.41.4.6 SegmentStructList_T AIRINV::FlightPeriodStruct::_segmentList

Definition at line 85 of file [FlightPeriodStruct.hpp](#).

Referenced by [addFareFamily\(\)](#), [addSegmentCabin\(\)](#), [buildSegments\(\)](#), and [describe\(\)](#).

24.41.4.7 bool AIRINV::FlightPeriodStruct::_legAlreadyDefined

Staging Leg (resp. Cabin) structure, gathering the result of the iteration on one leg (resp. cabin).

Definition at line 89 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

24.41.4.8 LegStruct AIRINV::FlightPeriodStruct::_itLeg

Definition at line 90 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

24.41.4.9 LegCabinStruct AIRINV::FlightPeriodStruct::_itLegCabin

Definition at line 91 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRINV::ScheduleParser-](#)

[Helper::storeCapacity::operator\(\)\(\)](#).

24.41.4.10 stdair::Date_T AIRINV::FlightPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 94 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), and [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#).

24.41.4.11 stdair::Date_T AIRINV::FlightPeriodStruct::_dateRangeEnd

Definition at line 95 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#).

24.41.4.12 unsigned int AIRINV::FlightPeriodStruct::_itYear

Definition at line 96 of file [FlightPeriodStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.41.4.13 unsigned int AIRINV::FlightPeriodStruct::_itMonth

Definition at line 97 of file [FlightPeriodStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.41.4.14 unsigned int AIRINV::FlightPeriodStruct::_itDay

Definition at line 98 of file [FlightPeriodStruct.hpp](#).

Referenced by [getDate\(\)](#).

24.41.4.15 int AIRINV::FlightPeriodStruct::_dateOffset

Definition at line 99 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), and [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#).

24.41.4.16 long AIRINV::FlightPeriodStruct::_itHours

Staging Time.

Definition at line 102 of file [FlightPeriodStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.41.4.17 long AIRINV::FlightPeriodStruct::_itMinutes

Definition at line 103 of file [FlightPeriodStruct.hpp](#).

Referenced by [getTime\(\)](#).

24.41.4.18 long AIRINV::FlightPeriodStruct::_itSeconds

Definition at line 104 of file [FlightPeriodStruct.hpp](#).

Referenced by [getTime\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), and [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#).

24.41.4.19 AirportList_T AIRINV::FlightPeriodStruct::_airportList

Staging Airport List (helper to derive the list of Segment structures).

Definition at line 108 of file [FlightPeriodStruct.hpp](#).

Referenced by [addAirport\(\)](#), [buildSegments\(\)](#), and [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

24.41.4.20 AirportOrderedList_T AIRINV::FlightPeriodStruct::_airportOrderedList

Definition at line 109 of file [FlightPeriodStruct.hpp](#).

Referenced by [addAirport\(\)](#), [buildSegments\(\)](#), and [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

24.41.4.21 bool AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 112 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#).

24.41.4.22 SegmentStruct AIRINV::FlightPeriodStruct::_itSegment

Definition at line 113 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#).

24.41.4.23 SegmentCabinStruct AIRINV::FlightPeriodStruct::_itSegmentCabin

Definition at line 114 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#).

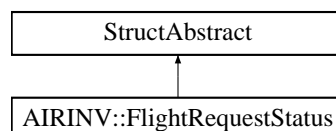
The documentation for this struct was generated from the following files:

- [airinv/bom/FlightPeriodStruct.hpp](#)
- [airinv/bom/FlightPeriodStruct.cpp](#)

24.42 AIRINV::FlightRequestStatus Struct Reference

```
#include <airinv/FlightRequestStatus.hpp>
```

Inheritance diagram for AIRINV::FlightRequestStatus:

**Public Types**

- enum [EN_FlightRequestStatus](#) { [OK](#) = 0, [NOT_FOUND](#), [INTERNAL_ERROR](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_FlightRequestStatus](#) `getCode () const`
- `const std::string` [describe \(\) const](#)
- [FlightRequestStatus](#) (`const EN_FlightRequestStatus &`)
- [FlightRequestStatus](#) (`const std::string &iCode`)

Static Public Member Functions

- `static const std::string &` [getLabel](#) (`const EN_FlightRequestStatus &`)
- `static const std::string &` [getCodeLabel](#) (`const EN_FlightRequestStatus &`)
- `static std::string` [describeLabels \(\)](#)

24.42.1 Detailed Description

Enumeration of flight type codes.

Definition at line 15 of file [FlightRequestStatus.hpp](#).

24.42.2 Member Enumeration Documentation

24.42.2.1 enum AIRINV::FlightRequestStatus::EN_FlightRequestStatus

Enumerator:

OK
NOT_FOUND
INTERNAL_ERROR
LAST_VALUE

Definition at line 17 of file [FlightRequestStatus.hpp](#).

24.42.3 Constructor & Destructor Documentation

24.42.3.1 AIRINV::FlightRequestStatus::FlightRequestStatus (const EN_FlightRequestStatus & iFlightRequestStatus)

Constructor.

Definition at line 25 of file [FlightRequestStatus.cpp](#).

24.42.3.2 AIRINV::FlightRequestStatus::FlightRequestStatus (const std::string & iCode)

Constructor.

Definition at line 30 of file [FlightRequestStatus.cpp](#).

References [describeLabels\(\)](#), [INTERNAL_ERROR](#), [LAST_VALUE](#), [NOT_FOUND](#), and [OK](#).

24.42.4 Member Function Documentation

24.42.4.1 const std::string & AIRINV::FlightRequestStatus::getLabel (const EN_FlightRequestStatus & iCode)
[static]

Get the label as a string.

Definition at line 58 of file [FlightRequestStatus.cpp](#).

24.42.4.2 `const std::string & AIRINV::FlightRequestStatus::getCodeLabel (const EN_FlightRequestStatus & iCode) [static]`

Get the label as a single char.

Definition at line 64 of file [FlightRequestStatus.cpp](#).

24.42.4.3 `std::string AIRINV::FlightRequestStatus::describeLabels () [static]`

List the labels.

Definition at line 69 of file [FlightRequestStatus.cpp](#).

References [LAST_VALUE](#).

Referenced by [FlightRequestStatus\(\)](#).

24.42.4.4 `FlightRequestStatus::EN_FlightRequestStatus AIRINV::FlightRequestStatus::getCode () const`

Get the enumerated value.

Definition at line 82 of file [FlightRequestStatus.cpp](#).

24.42.4.5 `const std::string AIRINV::FlightRequestStatus::describe () const`

Give a description of the structure (for display purposes).

Definition at line 87 of file [FlightRequestStatus.cpp](#).

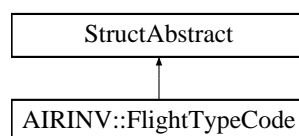
The documentation for this struct was generated from the following files:

- [airinv/FlightRequestStatus.hpp](#)
- [airinv/basic/FlightRequestStatus.cpp](#)

24.43 AIRINV::FlightTypeCode Struct Reference

```
#include <airinv/basic/FlightTypeCode.hpp>
```

Inheritance diagram for AIRINV::FlightTypeCode:



Public Types

- enum [EN_FlightTypeCode](#) { [DOMESTIC](#) = 0, [INTERNATIONAL](#), [GROUND_HANDLING](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_FlightTypeCode](#) [getCode](#) () const
- const std::string [describe](#) () const
- [FlightTypeCode](#) (const [EN_FlightTypeCode](#) &)
- [FlightTypeCode](#) (const std::string &iCode)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_FlightTypeCode](#) &)

- static const std::string & [getCodeLabel](#) (const [EN_FlightTypeCode](#) &)
- static std::string [describeLabels](#) ()

24.43.1 Detailed Description

Enumeration of flight type codes.

Definition at line 15 of file [FlightTypeCode.hpp](#).

24.43.2 Member Enumeration Documentation

24.43.2.1 enum AIRINV::FlightTypeCode::EN_FlightTypeCode

Enumerator:

DOMESTIC
INTERNATIONAL
GROUND_HANDLING
LAST_VALUE

Definition at line 17 of file [FlightTypeCode.hpp](#).

24.43.3 Constructor & Destructor Documentation

24.43.3.1 AIRINV::FlightTypeCode::FlightTypeCode (const [EN_FlightTypeCode](#) & *iFlightTypeCode*)

Constructor.

Definition at line 24 of file [FlightTypeCode.cpp](#).

24.43.3.2 AIRINV::FlightTypeCode::FlightTypeCode (const std::string & *iCode*)

Constructor.

Definition at line 29 of file [FlightTypeCode.cpp](#).

References [describeLabels\(\)](#), [DOMESTIC](#), [GROUND_HANDLING](#), [INTERNATIONAL](#), and [LAST_VALUE](#).

24.43.4 Member Function Documentation

24.43.4.1 const std::string & AIRINV::FlightTypeCode::getLabel (const [EN_FlightTypeCode](#) & *iCode*) [static]

Get the label as a string.

Definition at line 54 of file [FlightTypeCode.cpp](#).

24.43.4.2 const std::string & AIRINV::FlightTypeCode::getCodeLabel (const [EN_FlightTypeCode](#) & *iCode*) [static]

Get the label as a single char.

Definition at line 60 of file [FlightTypeCode.cpp](#).

24.43.4.3 std::string AIRINV::FlightTypeCode::describeLabels () [static]

List the labels.

Definition at line 65 of file [FlightTypeCode.cpp](#).

References [LAST_VALUE](#).

Referenced by [FlightTypeCode\(\)](#).

24.43.4.4 FlightTypeCode::EN_FlightTypeCode AIRINV::FlightTypeCode::getCode () const

Get the enumerated value.

Definition at line 77 of file [FlightTypeCode.cpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)\(\)](#).

24.43.4.5 const std::string AIRINV::FlightTypeCode::describe () const

Give a description of the structure (for display purposes).

Definition at line 82 of file [FlightTypeCode.cpp](#).

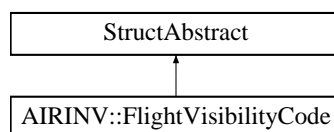
The documentation for this struct was generated from the following files:

- [airinv/basic/FlightTypeCode.hpp](#)
- [airinv/basic/FlightTypeCode.cpp](#)

24.44 AIRINV::FlightVisibilityCode Struct Reference

```
#include <airinv/basic/FlightVisibilityCode.hpp>
```

Inheritance diagram for AIRINV::FlightVisibilityCode:



Public Types

- enum [EN_FlightVisibilityCode](#) { [NORMAL](#) = 0, [HIDDEN](#), [PSEUDO](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_FlightVisibilityCode](#) [getCode](#) () const
- const std::string [describe](#) () const
- [FlightVisibilityCode](#) (const [EN_FlightVisibilityCode](#) &)
- [FlightVisibilityCode](#) (const std::string &iCode)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_FlightVisibilityCode](#) &)
- static const std::string & [getCodeLabel](#) (const [EN_FlightVisibilityCode](#) &)
- static std::string [describeLabels](#) ()

24.44.1 Detailed Description

Enumeration of flight visibility codes.

Definition at line 15 of file [FlightVisibilityCode.hpp](#).

24.44.2 Member Enumeration Documentation

24.44.2.1 enum AIRINV::FlightVisibilityCode::EN_FlightVisibilityCode

Enumerator:

NORMAL
HIDDEN
PSEUDO
LAST_VALUE

Definition at line 17 of file [FlightVisibilityCode.hpp](#).

24.44.3 Constructor & Destructor Documentation

24.44.3.1 AIRINV::FlightVisibilityCode::FlightVisibilityCode (const EN_FlightVisibilityCode & iFlightVisibilityCode)

Constructor.

Definition at line 25 of file [FlightVisibilityCode.cpp](#).

24.44.3.2 AIRINV::FlightVisibilityCode::FlightVisibilityCode (const std::string & iCode)

Constructor.

Definition at line 30 of file [FlightVisibilityCode.cpp](#).

References [describeLabels\(\)](#), [HIDDEN](#), [LAST_VALUE](#), [NORMAL](#), and [PSEUDO](#).

24.44.4 Member Function Documentation

24.44.4.1 const std::string & AIRINV::FlightVisibilityCode::getLabel (const EN_FlightVisibilityCode & iCode) [static]

Get the label as a string.

Definition at line 57 of file [FlightVisibilityCode.cpp](#).

24.44.4.2 const std::string & AIRINV::FlightVisibilityCode::getCodeLabel (const EN_FlightVisibilityCode & iCode) [static]

Get the label as a single char.

Definition at line 63 of file [FlightVisibilityCode.cpp](#).

24.44.4.3 std::string AIRINV::FlightVisibilityCode::describeLabels () [static]

List the labels.

Definition at line 68 of file [FlightVisibilityCode.cpp](#).

References [LAST_VALUE](#).

Referenced by [FlightVisibilityCode\(\)](#).

24.44.4.4 FlightVisibilityCode::EN_FlightVisibilityCode AIRINV::FlightVisibilityCode::getCode () const

Get the enumerated value.

Definition at line 81 of file [FlightVisibilityCode.cpp](#).

Referenced by [AIRINV::FlightDateStruct::describe\(\)](#), and [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#).

24.44.4.5 const std::string AIRINV::FlightVisibilityCode::describe () const

Give a description of the structure (for display purposes).

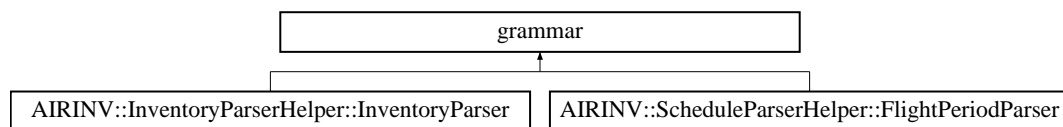
Definition at line 86 of file [FlightVisibilityCode.cpp](#).

The documentation for this struct was generated from the following files:

- [airinv/basic/FlightVisibilityCode.hpp](#)
- [airinv/basic/FlightVisibilityCode.cpp](#)

24.45 grammar Class Reference

Inheritance diagram for grammar:

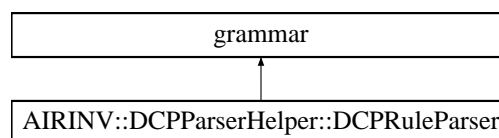


The documentation for this class was generated from the following file:

- [airinv/command/ScheduleParserHelper.hpp](#)

24.46 grammar Class Reference

Inheritance diagram for grammar:



The documentation for this class was generated from the following file:

- [airinv/command/vault/DCPParserHelper.hpp](#)

24.47 AIRINV::GuillotineBlockHelper Class Reference

```
#include <airinv/bom/GuillotineBlockHelper.hpp>
```

Static Public Member Functions

- static void [takeSnapshots](#) (stdair::GuillotineBlock &, const stdair::DateTime_T &)

24.47.1 Detailed Description

Class representing the actual business functions for an airline inventory.

Definition at line 22 of file [GuillotineBlockHelper.hpp](#).

24.47.2 Member Function Documentation

24.47.2.1 void AIRINV::GuillotineBlockHelper::takeSnapshots (stdair::GuillotineBlock & ioGuillotineBlock, const stdair::DateTime_T & iSnapshotTime) [static]

Take inventory snapshots.

Definition at line 27 of file [GuillotineBlockHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::updateAvailabilities\(\)](#).

The documentation for this class was generated from the following files:

- [airinv/bom/GuillotineBlockHelper.hpp](#)
- [airinv/bom/GuillotineBlockHelper.cpp](#)

24.48 AIRINV::header Struct Reference

```
#include <airinv/server/header.hpp>
```

Public Attributes

- std::string [name](#)
- std::string [value](#)

24.48.1 Detailed Description

Header structure.

Definition at line 13 of file [header.hpp](#).

24.48.2 Member Data Documentation

24.48.2.1 std::string AIRINV::header::name

Definition at line 14 of file [header.hpp](#).

24.48.2.2 std::string AIRINV::header::value

Definition at line 15 of file [header.hpp](#).

The documentation for this struct was generated from the following file:

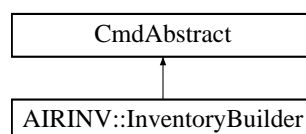
- [airinv/server/header.hpp](#)

24.49 AIRINV::InventoryBuilder Class Reference

Class handling the generation / instantiation of the Inventory BOM.

```
#include <airinv/command/InventoryBuilder.hpp>
```

Inheritance diagram for AIRINV::InventoryBuilder:



Friends

- struct [InventoryParserHelper::doEndFlightDate](#)

24.49.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

Definition at line 43 of file [InventoryBuilder.hpp](#).

24.49.2 Friends And Related Function Documentation

24.49.2.1 friend struct [InventoryParserHelper::doEndFlightDate](#) [friend]

Only the following class may use methods of [InventoryBuilder](#). Indeed, as those methods build the BOM, it is not good to expose them publicly.

Definition at line 49 of file [InventoryBuilder.hpp](#).

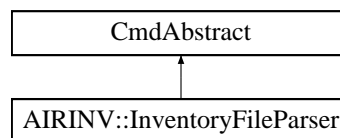
The documentation for this class was generated from the following files:

- [airinv/command/InventoryBuilder.hpp](#)
- [airinv/command/InventoryBuilder.cpp](#)

24.50 AIRINV::InventoryFileParser Class Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryFileParser:



Public Member Functions

- [InventoryFileParser](#) (stdair::BomRoot &, const stdair::Filename_T & iInventoryInputFilename)
- bool [buildInventory](#) ()

24.50.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 500 of file [InventoryParserHelper.hpp](#).

24.50.2 Constructor & Destructor Documentation

24.50.2.1 AIRINV::InventoryFileParser::InventoryFileParser (stdair::BomRoot & , const stdair::Filename.T & iInventoryInputFilename)

Constructor.

Definition at line 1092 of file [InventoryParserHelper.cpp](#).

24.50.3 Member Function Documentation

24.50.3.1 bool AIRINV::InventoryFileParser::buildInventory ()

Parse the inventory input file.

Definition at line 1116 of file [InventoryParserHelper.cpp](#).

Referenced by [AIRINV::InventoryParser::buildInventory\(\)](#).

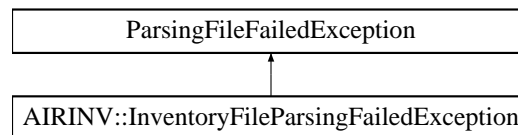
The documentation for this class was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.51 AIRINV::InventoryFileParsingFailedException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::InventoryFileParsingFailedException:



Public Member Functions

- [InventoryFileParsingFailedException](#) (const std::string &iWhat)

24.51.1 Detailed Description

The inventory input file can not be parsed.

Definition at line 27 of file [AIRINV_Types.hpp](#).

24.51.2 Constructor & Destructor Documentation

24.51.2.1 AIRINV::InventoryFileParsingFailedException::InventoryFileParsingFailedException (const std::string & iWhat) [inline]

Constructor.

Definition at line 33 of file [AIRINV_Types.hpp](#).

The documentation for this class was generated from the following file:

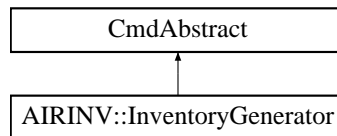
- [airinv/AIRINV_Types.hpp](#)

24.52 AIRINV::InventoryGenerator Class Reference

Class handling the generation / instantiation of the Inventory BOM.

```
#include <airinv/command/InventoryGenerator.hpp>
```

Inheritance diagram for AIRINV::InventoryGenerator:



Friends

- class [FlightPeriodFileParser](#)
- class [FFFlightPeriodFileParser](#)
- struct [ScheduleParserHelper::doEndFlight](#)
- class [ScheduleParser](#)

24.52.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

Definition at line 42 of file [InventoryGenerator.hpp](#).

24.52.2 Friends And Related Function Documentation

24.52.2.1 friend class [FlightPeriodFileParser](#) [friend]

Only the following class may use methods of [InventoryGenerator](#). Indeed, as those methods build the BOM, it is not good to expose them publicly.

Definition at line 48 of file [InventoryGenerator.hpp](#).

24.52.2.2 friend class [FFFlightPeriodFileParser](#) [friend]

Definition at line 49 of file [InventoryGenerator.hpp](#).

24.52.2.3 friend struct [ScheduleParserHelper::doEndFlight](#) [friend]

Definition at line 50 of file [InventoryGenerator.hpp](#).

24.52.2.4 friend class [ScheduleParser](#) [friend]

Definition at line 51 of file [InventoryGenerator.hpp](#).

The documentation for this class was generated from the following files:

- [airinv/command/InventoryGenerator.hpp](#)
- [airinv/command/InventoryGenerator.cpp](#)

24.53 AIRINV::InventoryHelper Class Reference

```
#include <airinv/bom/InventoryHelper.hpp>
```

Static Public Member Functions

- static void [fillFromRouting](#) (const stdair::Inventory &)
- static void [calculateAvailability](#) (const stdair::Inventory &, const std::string &, stdair::TravelSolutionStruct &)
- static void [getYieldAndBidPrice](#) (const stdair::Inventory &, const std::string &, stdair::TravelSolutionStruct &)
- static bool [sell](#) (stdair::Inventory &, const std::string & iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)

- static bool [cancel](#) (stdair::Inventory &, const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- static void [takeSnapshots](#) (const stdair::Inventory &, const stdair::DateTime_T &)

24.53.1 Detailed Description

Class representing the actual business functions for an airline inventory.

Definition at line 22 of file [InventoryHelper.hpp](#).

24.53.2 Member Function Documentation

24.53.2.1 void [AIRINV::InventoryHelper::fillFromRouting](#) (const stdair::Inventory & *ilInventory*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 28 of file [InventoryHelper.cpp](#).

24.53.2.2 void [AIRINV::InventoryHelper::calculateAvailability](#) (const stdair::Inventory & *ilInventory*, const std::string & *iFullSegmentDateKey*, stdair::TravelSolutionStruct & *ioTravelSolution*) [static]

Compute the availability for the given travel solution.

Definition at line 44 of file [InventoryHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::updateAvailabilities\(\)](#).

24.53.2.3 void [AIRINV::InventoryHelper::getYieldAndBidPrice](#) (const stdair::Inventory & *ilInventory*, const std::string & *iFullSegmentDateKey*, stdair::TravelSolutionStruct & *ioTravelSolution*) [static]

Get yield and bid price information for the given travel solution.

Definition at line 97 of file [InventoryHelper.cpp](#).

24.53.2.4 bool [AIRINV::InventoryHelper::sell](#) (stdair::Inventory & *ioInventory*, const std::string & *iSegmentDateKey*, const stdair::ClassCode_T & *iClassCode*, const stdair::PartySize_T & *iPartySize*) [static]

Make a sale with the given travel solution.

Definition at line 239 of file [InventoryHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::updateFromReservation\(\)](#).

24.53.2.5 bool [AIRINV::InventoryHelper::cancel](#) (stdair::Inventory & *ioInventory*, const std::string & *iSegmentDateKey*, const stdair::ClassCode_T & *iClassCode*, const stdair::PartySize_T & *iPartySize*) [static]

Make a cancellation.

Definition at line 295 of file [InventoryHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::updateFromReservation\(\)](#).

24.53.2.6 void [AIRINV::InventoryHelper::takeSnapshots](#) (const stdair::Inventory & *ilInventory*, const stdair::DateTime_T & *iSnapshotTime*) [static]

Take inventory snapshots.

Definition at line 351 of file [InventoryHelper.cpp](#).

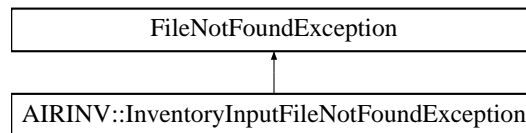
The documentation for this class was generated from the following files:

- [airinv/bom/InventoryHelper.hpp](#)
- [airinv/bom/InventoryHelper.cpp](#)

24.54 AIRINV::InventoryInputFileNotFoundException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::InventoryInputFileNotFoundException:



Public Member Functions

- [InventoryInputFileNotFoundException](#) (const std::string &iWhat)

24.54.1 Detailed Description

The inventory input file can not be found or opened.

Definition at line 66 of file [AIRINV_Types.hpp](#).

24.54.2 Constructor & Destructor Documentation

24.54.2.1 [AIRINV::InventoryInputFileNotFoundException::InventoryInputFileNotFoundException \(const std::string &iWhat \)](#) `[inline]`

Constructor.

Definition at line 71 of file [AIRINV_Types.hpp](#).

The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.55 AIRINV::InventoryManager Class Reference

```
#include <airinv/command/InventoryManager.hpp>
```

Static Public Member Functions

- static void [createDirectAccesses](#) (const stdair::BomRoot &)
- static void [createDirectAccesses](#) (stdair::Inventory &)
- static void [createDirectAccesses](#) (stdair::FlightDate &)
- static void [createDirectAccesses](#) (stdair::SegmentDate &)
- static void [buildSimilarSegmentCabinSets](#) (const stdair::BomRoot &)
- static void [buildSimilarSegmentCabinSets](#) (stdair::Inventory &)
- static void [buildGuillotineBlock](#) (stdair::Inventory &, const stdair::GuillotineNumber_T &, const [DepartureDate-SegmentCabinMap_T](#) &)
- static void [setDefaultBidPriceVector](#) (stdair::BomRoot &)
- static void [setDefaultBidPriceVector](#) (stdair::Inventory &)

Friends

- class [AIRINV_Master_Service](#)
- class [AIRINV_Service](#)

24.55.1 Detailed Description

Command wrapping the travel request process.

Definition at line 34 of file [InventoryManager.hpp](#).

24.55.2 Member Function Documentation

24.55.2.1 void AIRINV::InventoryManager::createDirectAccesses (const stdair::BomRoot & *iBomRoot*)
[static]

Create the direct accesses within the inventories such as links between leg-date and segment-date, ect.

Definition at line 717 of file [InventoryManager.cpp](#).

References [AIRINV::BomRootHelper::fillFromRouting\(\)](#).

Referenced by [AIRINV::InventoryParser::buildInventory\(\)](#), [createDirectAccesses\(\)](#), and [AIRINV::ScheduleParser::generateInventories\(\)](#).

24.55.2.2 void AIRINV::InventoryManager::createDirectAccesses (stdair::Inventory & *ioInventory*) [static]

Definition at line 737 of file [InventoryManager.cpp](#).

References [createDirectAccesses\(\)](#).

24.55.2.3 void AIRINV::InventoryManager::createDirectAccesses (stdair::FlightDate & *ioFlightDate*) [static]

Definition at line 755 of file [InventoryManager.cpp](#).

References [createDirectAccesses\(\)](#).

24.55.2.4 void AIRINV::InventoryManager::createDirectAccesses (stdair::SegmentDate & *ioSegmentDate*)
[static]

Definition at line 824 of file [InventoryManager.cpp](#).

24.55.2.5 void AIRINV::InventoryManager::buildSimilarSegmentCabinSets (const stdair::BomRoot & *iBomRoot*)
[static]

Build the similar segment-cabin sets and the corresponding guillotine blocks for snapshots and other data.

Definition at line 890 of file [InventoryManager.cpp](#).

Referenced by [AIRINV::AIRINV_Service::buildSampleBom\(\)](#), and [AIRINV::ScheduleParser::generateInventories\(\)](#).

24.55.2.6 void AIRINV::InventoryManager::buildSimilarSegmentCabinSets (stdair::Inventory & *ioInventory*)
[static]

Definition at line 906 of file [InventoryManager.cpp](#).

References [buildGuillotineBlock\(\)](#).

24.55.2.7 void AIRINV::InventoryManager::buildGuillotineBlock (stdair::Inventory & *ioInventory*, const stdair::GuillotineNumber_T & *iGuillotineNumber*, const DepartureDateSegmentCabinMap_T & *iDDSCMap*)
[static]

Definition at line 981 of file [InventoryManager.cpp](#).

Referenced by [buildSimilarSegmentCabinSets\(\)](#).

24.55.2.8 void AIRINV::InventoryManager::setDefaultBidPriceVector (stdair::BomRoot & *ioBomRoot*)
[static]

Bid price vectors initialisation

Definition at line 596 of file [InventoryManager.cpp](#).

Referenced by [AIRINV::ScheduleParser::generateInventories\(\)](#).

24.55.2.9 **void AIRINV::InventoryManager::setDefaultBidPriceVector** ([stdair::Inventory](#) & *ioInventory*)
[static]

Definition at line 628 of file [InventoryManager.cpp](#).

24.55.3 Friends And Related Function Documentation

24.55.3.1 **friend class AIRINV_Master_Service** [friend]

Definition at line 35 of file [InventoryManager.hpp](#).

24.55.3.2 **friend class AIRINV_Service** [friend]

Definition at line 36 of file [InventoryManager.hpp](#).

The documentation for this class was generated from the following files:

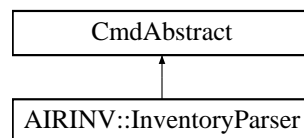
- [airinv/command/InventoryManager.hpp](#)
- [airinv/command/InventoryManager.cpp](#)

24.56 AIRINV::InventoryParser Class Reference

Class wrapping the parser entry point.

```
#include <airinv/command/InventoryParser.hpp>
```

Inheritance diagram for AIRINV::InventoryParser:



Static Public Member Functions

- static void [buildInventory](#) (const [stdair::Filename_T](#) & *InventoryFilename*, [stdair::BomRoot](#) &)

24.56.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 21 of file [InventoryParser.hpp](#).

24.56.2 Member Function Documentation

24.56.2.1 **void AIRINV::InventoryParser::buildInventory** (const [stdair::Filename_T](#) & *InventoryFilename*,
[stdair::BomRoot](#) & *ioBomRoot*) [static]

Parses the CSV file describing an airline inventory, and generates the corresponding data model in memory. It can then be used, for instance, in a simulator.

Parameters

<i>const</i>	stdair::Filename_T& The file-name of the CSV-formatted inventory input file.
<i>stdair::Bom-Root&</i>	Root of the BOM tree.

Definition at line 20 of file [InventoryParser.cpp](#).

References [AIRINV::InventoryFileParser::buildInventory\(\)](#), and [AIRINV::InventoryManager::createDirectAccesses\(\)](#).

Referenced by [AIRINV::AIRINV_Service::parseAndLoad\(\)](#).

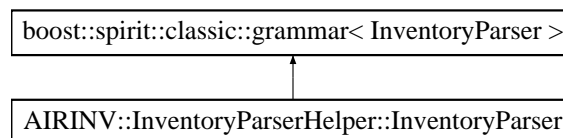
The documentation for this class was generated from the following files:

- [airinv/command/InventoryParser.hpp](#)
- [airinv/command/InventoryParser.cpp](#)

24.57 AIRINV::InventoryParserHelper::InventoryParser Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::InventoryParser:



Classes

- struct [definition](#)

Public Member Functions

- [InventoryParser](#) (stdair::BomRoot &, [FlightDateStruct](#) &, unsigned int &)

Public Attributes

- stdair::BomRoot & [_bomRoot](#)
- [FlightDateStruct](#) & [_flightDate](#)
- unsigned int & [_nbOfFlights](#)

24.57.1 Detailed Description

FlightDepDate; 2010-02-08; SIN; BKK; L; 10.0; 1.0;

Grammar: FlightDate ::= FlightDepDate ';' Origin ';' Destination EndOfFlightDate FlightDepDate ::= date EndOf-FlightDate ::= ';' Grammar for the inventory parser.

Definition at line 454 of file [InventoryParserHelper.hpp](#).

24.57.2 Constructor & Destructor Documentation

24.57.2.1 AIRINV::InventoryParserHelper::InventoryParser::InventoryParser (stdair::BomRoot & *ioBomRoot*, [FlightDateStruct](#) & *ioFlightDate*, unsigned int & *ioNbOfFlights*)

Definition at line 862 of file [InventoryParserHelper.cpp](#).

24.57.3 Member Data Documentation

24.57.3.1 `stdair::BomRoot& AIRINV::InventoryParserHelper::InventoryParser::_bomRoot`

Definition at line 482 of file [InventoryParserHelper.hpp](#).

24.57.3.2 `FlightDateStruct& AIRINV::InventoryParserHelper::InventoryParser::_flightDate`

Definition at line 483 of file [InventoryParserHelper.hpp](#).

24.57.3.3 `unsigned int& AIRINV::InventoryParserHelper::InventoryParser::_nbOfFlights`

Definition at line 484 of file [InventoryParserHelper.hpp](#).

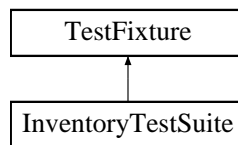
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.58 InventoryTestSuite Class Reference

```
#include <test/airinv/InventoryTestSuite.hpp>
```

Inheritance diagram for InventoryTestSuite:



Public Member Functions

- void [simpleInventory](#) ()
- [InventoryTestSuite](#) ()

Protected Attributes

- `std::stringstream` [_describeKey](#)

24.58.1 Detailed Description

Utility class for CppUnit-based testing.

Definition at line 7 of file [InventoryTestSuite.hpp](#).

24.58.2 Constructor & Destructor Documentation

24.58.2.1 `InventoryTestSuite::InventoryTestSuite ()`

Test some error detection functionalities. Constructor.

24.58.3 Member Function Documentation

24.58.3.1 void InventoryTestSuite::simpleInventory ()

Test a simple inventory functionality.

24.58.4 Member Data Documentation

24.58.4.1 std::stringstream InventoryTestSuite::_describeKey [protected]

Definition at line 28 of file [InventoryTestSuite.hpp](#).

The documentation for this class was generated from the following file:

- [test/airinv/InventoryTestSuite.hpp](#)

24.59 AIRINV::LegCabinHelper Class Reference

```
#include <airinv/bom/LegCabinHelper.hpp>
```

24.59.1 Detailed Description

Class representing the actual business functions for an airline leg-cabin.

Definition at line 16 of file [LegCabinHelper.hpp](#).

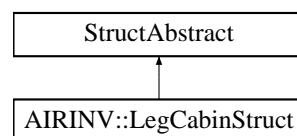
The documentation for this class was generated from the following file:

- [airinv/bom/LegCabinHelper.hpp](#)

24.60 AIRINV::LegCabinStruct Struct Reference

```
#include <airinv/bom/LegCabinStruct.hpp>
```

Inheritance diagram for AIRINV::LegCabinStruct:



Public Member Functions

- void [fill](#) (stdair::LegCabin &) const
- const std::string [describe](#) () const

Public Attributes

- stdair::CabinCode_T [_cabinCode](#)
- stdair::CabinCapacity_T [_saleableCapacity](#)
- stdair::CapacityAdjustment_T [_adjustment](#)
- stdair::CapacityAdjustment_T [_dcsRegrade](#)
- stdair::AuthorizationLevel_T [_au](#)

- [stdair::Availability_T _avPool](#)
- [stdair::UPR_T _upr](#)
- [stdair::NbOfBookings_T _nbOfBookings](#)
- [stdair::Availability_T _nav](#)
- [stdair::Availability_T _gav](#)
- [stdair::OverbookingRate_T _acp](#)
- [stdair::NbOfBookings_T _etb](#)
- [stdair::NbOfBookings_T _staffNbOfBookings](#)
- [stdair::NbOfBookings_T _wINbOfBookings](#)
- [stdair::NbOfBookings_T _groupNbOfBookings](#)
- [BucketStructList_T _bucketList](#)

24.60.1 Detailed Description

Utility Structure for the parsing of LegCabin details.

Definition at line 24 of file [LegCabinStruct.hpp](#).

24.60.2 Member Function Documentation

24.60.2.1 void AIRINV::LegCabinStruct::fill (stdair::LegCabin & *ioLegCabin*) const

Fill the LegCabin objects with the attributes of the [LegCabinStruct](#).

Definition at line 38 of file [LegCabinStruct.cpp](#).

References [_saleableCapacity](#).

24.60.2.2 const std::string AIRINV::LegCabinStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 15 of file [LegCabinStruct.cpp](#).

References [_acp](#), [_adjustment](#), [_au](#), [_avPool](#), [_bucketList](#), [_cabinCode](#), [_dcsRegrade](#), [_etb](#), [_gav](#), [_groupNbOfBookings](#), [_nav](#), [_nbOfBookings](#), [_saleableCapacity](#), [_staffNbOfBookings](#), [_upr](#), [_wINbOfBookings](#), and [AIRINV::BucketStruct::describe\(\)](#).

Referenced by [AIRINV::LegStruct::describe\(\)](#).

24.60.3 Member Data Documentation

24.60.3.1 stdair::CabinCode_T AIRINV::LegCabinStruct::_cabinCode

Definition at line 26 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#).

24.60.3.2 stdair::CabinCapacity_T AIRINV::LegCabinStruct::_saleableCapacity

Definition at line 27 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), and [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)\(\)](#).

24.60.3.3 stdair::CapacityAdjustment_T AIRINV::LegCabinStruct::_adjustment

Definition at line 28 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.4 stdair::CapacityAdjustment_T AIRINV::LegCabinStruct::_dcsRegrade

Definition at line 29 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.5 stdair::AuthorizationLevel_T AIRINV::LegCabinStruct::_au

Definition at line 30 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#).

24.60.3.6 stdair::Availability_T AIRINV::LegCabinStruct::_avPool

Definition at line 31 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.7 stdair::UPR_T AIRINV::LegCabinStruct::_upr

Definition at line 32 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#).

24.60.3.8 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_nbOfBookings

Definition at line 33 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#).

24.60.3.9 stdair::Availability_T AIRINV::LegCabinStruct::_nav

Definition at line 34 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#).

24.60.3.10 stdair::Availability_T AIRINV::LegCabinStruct::_gav

Definition at line 35 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#).

24.60.3.11 stdair::OverbookingRate_T AIRINV::LegCabinStruct::_acp

Definition at line 36 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#).

24.60.3.12 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_etb

Definition at line 37 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#).

24.60.3.13 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_staffNbOfBookings

Definition at line 38 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.14 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_wINbOfBookings

Definition at line 39 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.15 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_groupNbOfBookings

Definition at line 40 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#).

24.60.3.16 BucketStructList_T AIRINV::LegCabinStruct::_bucketList

Definition at line 41 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

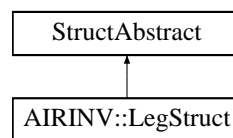
The documentation for this struct was generated from the following files:

- [airinv/bom/LegCabinStruct.hpp](#)
- [airinv/bom/LegCabinStruct.cpp](#)

24.61 AIRINV::LegStruct Struct Reference

```
#include <airinv/bom/LegStruct.hpp>
```

Inheritance diagram for AIRINV::LegStruct:



Public Member Functions

- void [fill](#) (const stdair::Date_T &iRefDate, stdair::LegDate &) const
- void [fill](#) (stdair::LegDate &) const
- const std::string [describe](#) () const
- [LegStruct](#) ()

Public Attributes

- stdair::AirportCode_T [_boardingPoint](#)
- stdair::DateOffset_T [_boardingDateOffset](#)
- stdair::Date_T [_boardingDate](#)
- stdair::Duration_T [_boardingTime](#)
- stdair::AirportCode_T [_offPoint](#)
- stdair::DateOffset_T [_offDateOffset](#)
- stdair::Date_T [_offDate](#)
- stdair::Duration_T [_offTime](#)
- stdair::Duration_T [_elapsed](#)
- [LegCabinStructList_T](#) [_cabinList](#)

24.61.1 Detailed Description

Utility Structure for the parsing of Leg structures.

Definition at line 24 of file [LegStruct.hpp](#).

24.61.2 Constructor & Destructor Documentation

24.61.2.1 AIRINV::LegStruct::LegStruct ()

Default Constructor.

Definition at line 16 of file [LegStruct.cpp](#).

24.61.3 Member Function Documentation

24.61.3.1 void AIRINV::LegStruct::fill (const stdair::Date_T & iRefDate, stdair::LegDate & ioLegDate) const

Fill the LegDate objects with the attributes of the [LegStruct](#).

The given reference date corresponds to the date of the FlightDate. Indeed, each Leg gets date off-sets, when compared to that (reference) flight-date, both for the boarding date and for the off date.

Definition at line 41 of file [LegStruct.cpp](#).

References [_boardingDateOffset](#), [_boardingTime](#), [_elapsed](#), [_offDateOffset](#), [_offPoint](#), and [_offTime](#).

24.61.3.2 void AIRINV::LegStruct::fill (stdair::LegDate & ioLegDate) const

Fill the LegDate objects with the attributes of the [LegStruct](#).

Definition at line 58 of file [LegStruct.cpp](#).

References [_boardingTime](#), [_elapsed](#), [_offDate](#), [_offPoint](#), and [_offTime](#).

24.61.3.3 const std::string AIRINV::LegStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 21 of file [LegStruct.cpp](#).

References [_boardingDate](#), [_boardingPoint](#), [_boardingTime](#), [_cabinList](#), [_elapsed](#), [_offDate](#), [_offPoint](#), [_offTime](#), and [AIRINV::LegCabinStruct::describe\(\)](#).

Referenced by [AIRINV::FlightPeriodStruct::describe\(\)](#), and [AIRINV::FlightDateStruct::describe\(\)](#).

24.61.4 Member Data Documentation

24.61.4.1 stdair::AirportCode_T AIRINV::LegStruct::_boardingPoint

Definition at line 26 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), and [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#).

24.61.4.2 stdair::DateOffset_T AIRINV::LegStruct::_boardingDateOffset

Definition at line 27 of file [LegStruct.hpp](#).

Referenced by [fill\(\)](#), and [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#).

24.61.4.3 stdair::Date_T AIRINV::LegStruct::_boardingDate

Definition at line 28 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)\(\)](#).

24.61.4.4 stdair::Duration_T AIRINV::LegStruct::_boardingTime

Definition at line 29 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#).

24.61.4.5 `stdair::AirportCode_T AIRINV::LegStruct::_offPoint`

Definition at line 30 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#).

24.61.4.6 `stdair::DateOffset_T AIRINV::LegStruct::_offDateOffset`

Definition at line 31 of file [LegStruct.hpp](#).

Referenced by [fill\(\)](#), and [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

24.61.4.7 `stdair::Date_T AIRINV::LegStruct::_offDate`

Definition at line 32 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#).

24.61.4.8 `stdair::Duration_T AIRINV::LegStruct::_offTime`

Definition at line 33 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#).

24.61.4.9 `stdair::Duration_T AIRINV::LegStruct::_elapsed`

Definition at line 34 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

24.61.4.10 `LegCabinStructList_T AIRINV::LegStruct::_cabinList`

Definition at line 35 of file [LegStruct.hpp](#).

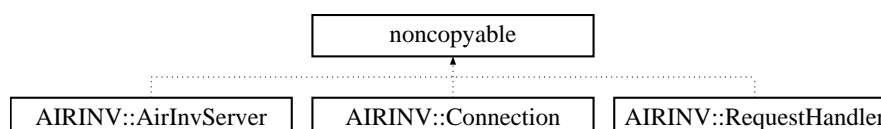
Referenced by [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/bom/LegStruct.hpp](#)
- [airinv/bom/LegStruct.cpp](#)

24.62 noncopyable Class Reference

Inheritance diagram for noncopyable:

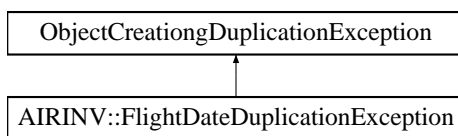


The documentation for this class was generated from the following file:

- [airinv/server/Connection.hpp](#)

24.63 ObjectCreationgDuplicationException Class Reference

Inheritance diagram for ObjectCreationgDuplicationException:

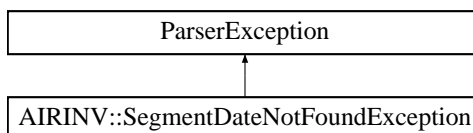


The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.64 ParserException Class Reference

Inheritance diagram for ParserException:



The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.65 AIRINV::InventoryParserHelper::ParserSemanticAction Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::ParserSemanticAction:

[illegible]

Public Member Functions

- ParserSemanticAction (FlightDateStruct &)

Public Attributes

- FlightDateStruct & _flightDate

24.65.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Inventory Parser.

Definition at line 29 of file InventoryParserHelper.hpp.

24.65.2 Constructor & Destructor Documentation

24.65.2.1 **AIRINV::InventoryParserHelper::ParserSemanticAction::ParserSemanticAction (FlightDateStruct & ioFlightDate)**

Actor Constructor.

Definition at line 26 of file InventoryParserHelper.cpp.

24.65.3 Member Data Documentation

24.65.3.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

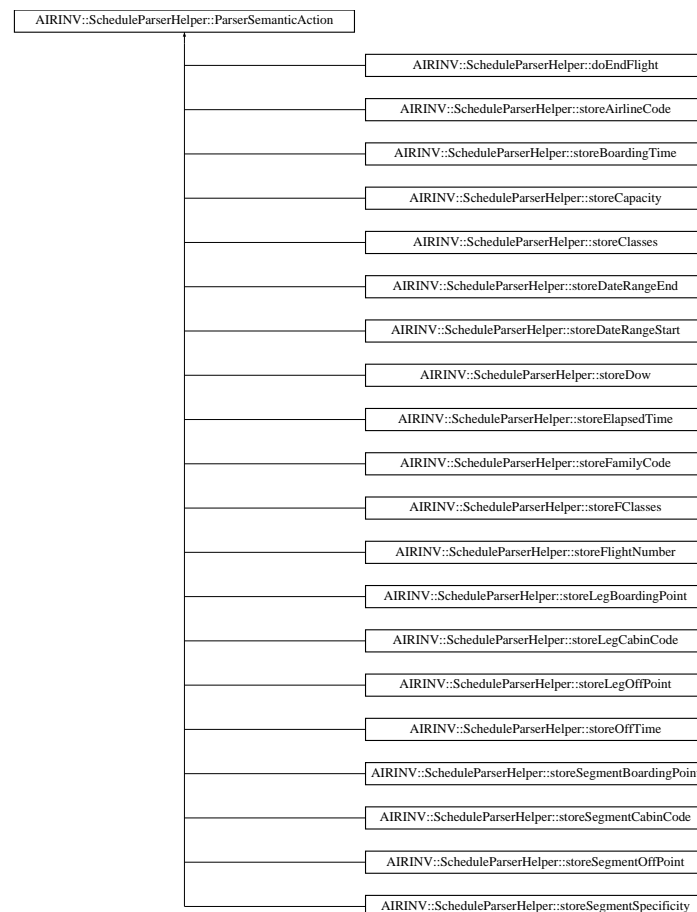
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.66 AIRINV::ScheduleParserHelper::ParserSemanticAction Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::ParserSemanticAction:



Public Member Functions

- [ParserSemanticAction](#) ([FlightPeriodStruct](#) &)

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.66.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

Definition at line 29 of file [ScheduleParserHelper.hpp](#).

24.66.2 Constructor & Destructor Documentation

24.66.2.1 AIRINV::ScheduleParserHelper::ParserSemanticAction::ParserSemanticAction ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 27 of file [ScheduleParserHelper.cpp](#).

24.66.3 Member Data Documentation

24.66.3.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

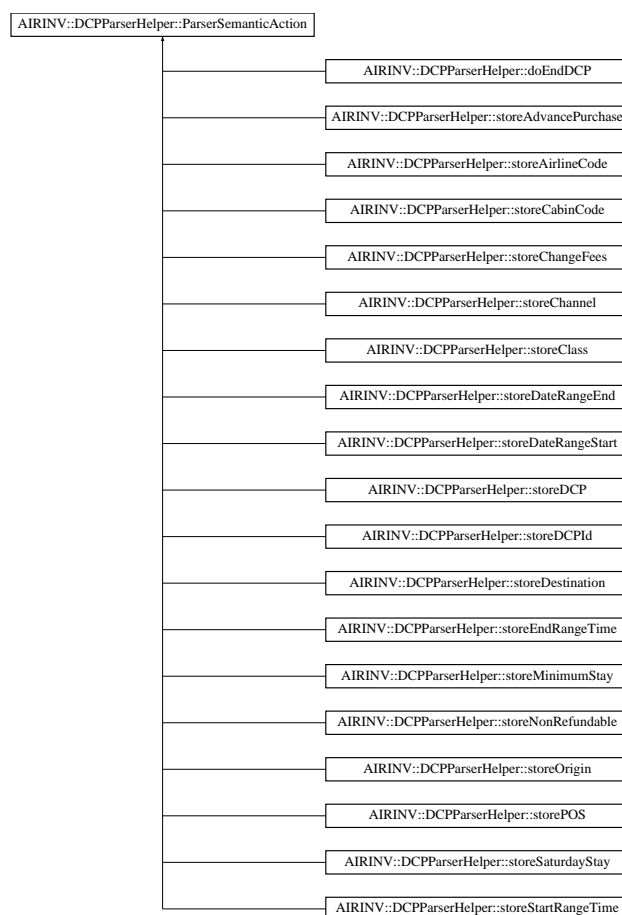
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.67 AIRINV::DCPParserHelper::ParserSemanticAction Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::ParserSemanticAction:



Public Member Functions

- [ParserSemanticAction](#) (DCPRuleStruct &)

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.67.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the DCP Parser.

Definition at line 30 of file [DCPParserHelper.hpp](#).

24.67.2 Constructor & Destructor Documentation

24.67.2.1 AIRINV::DCPParserHelper::ParserSemanticAction::ParserSemanticAction (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 25 of file [DCPParserHelper.cpp](#).

24.67.3 Member Data Documentation

24.67.3.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

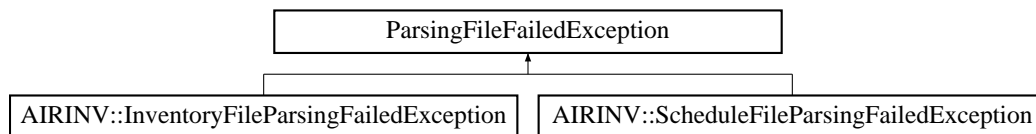
Referenced by [AIRINV::DCPParserHelper::storeDCPID::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.68 ParsingFileFailedException Class Reference

Inheritance diagram for ParsingFileFailedException:



The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.69 AIRINV::Reply Struct Reference

```
#include <airinv/server/Reply.hpp>
```

Public Member Functions

- `std::vector< boost::asio::const_buffer > to_buffers ()`

Public Attributes

- `FlightRequestStatus::EN_FlightRequestStatus _status`
- `std::string content`

24.69.1 Detailed Description

A reply to be sent to a client.

Definition at line 18 of file [Reply.hpp](#).

24.69.2 Member Function Documentation

24.69.2.1 `std::vector< boost::asio::const_buffer > AIRINV::Reply::to_buffers ()`

Convert the reply into a vector of buffers. The buffers do not own the underlying memory blocks, therefore the reply object must remain valid and not be changed until the write operation has completed.

Definition at line 15 of file [Reply.cpp](#).

References [content](#).

24.69.3 Member Data Documentation

24.69.3.1 `FlightRequestStatus::EN_FlightRequestStatus AIRINV::Reply::_status`

Status.

Definition at line 20 of file [Reply.hpp](#).

Referenced by [AIRINV::RequestHandler::handleRequest\(\)](#).

24.69.3.2 `std::string AIRINV::Reply::content`

The content to be sent in the reply.

Definition at line 23 of file [Reply.hpp](#).

Referenced by [AIRINV::RequestHandler::handleRequest\(\)](#), and [to_buffers\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/server/Reply.hpp](#)
- [airinv/server/Reply.cpp](#)

24.70 AIRINV::Request Struct Reference

```
#include <airinv/server/Request.hpp>
```

Public Member Functions

- `bool parseFlightDate ()`

Public Attributes

- `std::string _flightDetails`
- `stdair::AirlineCode_T _airlineCode`
- `stdair::FlightNumber_T _flightNumber`
- `stdair::Date_T _departureDate`

24.70.1 Detailed Description

A request received from a client.

Definition at line 18 of file [Request.hpp](#).

24.70.2 Member Function Documentation

24.70.2.1 bool AIRINV::Request::parseFlightDate ()

Parse the incoming request.

Expected requested is of the form: <airline_code>,<flight_number>,<flight_date>, where date format is YYYY-MM-DD. For instance: BA,341,2010-09-20.

Definition at line 12 of file [Request.cpp](#).

References [_airlineCode](#), [_departureDate](#), and [_flightNumber](#).

Referenced by [AIRINV::RequestHandler::handleRequest\(\)](#).

24.70.3 Member Data Documentation

24.70.3.1 std::string AIRINV::Request::_flightDetails

String as it comes from the connected client.

Definition at line 29 of file [Request.hpp](#).

Referenced by [AIRINV::RequestHandler::handleRequest\(\)](#).

24.70.3.2 stdair::AirlineCode.T AIRINV::Request::_airlineCode

Parsed airline code.

Definition at line 31 of file [Request.hpp](#).

Referenced by [parseFlightDate\(\)](#).

24.70.3.3 stdair::FlightNumber.T AIRINV::Request::_flightNumber

Parsed flight number.

Definition at line 33 of file [Request.hpp](#).

Referenced by [parseFlightDate\(\)](#).

24.70.3.4 stdair::Date.T AIRINV::Request::_departureDate

Parsed departure date.

Definition at line 35 of file [Request.hpp](#).

Referenced by [parseFlightDate\(\)](#).

The documentation for this struct was generated from the following files:

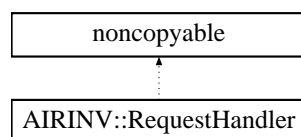
- [airinv/server/Request.hpp](#)
- [airinv/server/Request.cpp](#)

24.71 AIRINV::RequestHandler Class Reference

The common handler for all incoming requests.

```
#include <airinv/server/RequestHandler.hpp>
```

Inheritance diagram for AIRINV::RequestHandler:



Public Member Functions

- [RequestHandler](#) (const stdair::AirlineCode_T &)
- bool [handleRequest](#) ([Request](#) &, [Reply](#) &) const

24.71.1 Detailed Description

The common handler for all incoming requests.

Definition at line 28 of file [RequestHandler.hpp](#).

24.71.2 Constructor & Destructor Documentation

24.71.2.1 AIRINV::RequestHandler::RequestHandler (const stdair::AirlineCode_T & iAirlineCode)

Constructor.

Parameters

<i>const</i>	stdair::AirlineCode_T& Airline code of the inventory owner.
--------------	---

Definition at line 20 of file [RequestHandler.cpp](#).

24.71.3 Member Function Documentation

24.71.3.1 bool AIRINV::RequestHandler::handleRequest ([Request](#) & ioRequest, [Reply](#) & ioReply) const

Handle a request and produce a reply.

Definition at line 26 of file [RequestHandler.cpp](#).

References [AIRINV::Request::_flightDetails](#), [AIRINV::Reply::_status](#), [AIRINV::Reply::content](#), [AIRINV::FlightRequestStatus::INTERNAL_ERROR](#), [AIRINV::FlightRequestStatus::OK](#), and [AIRINV::Request::parseFlightDate\(\)](#).

The documentation for this class was generated from the following files:

- [airinv/server/RequestHandler.hpp](#)
- [airinv/server/RequestHandler.cpp](#)

24.72 AIRINV::RequestParser Class Reference

Parser for incoming requests.

```
#include <airinv/server/RequestParser.hpp>
```

Public Member Functions

- [RequestParser](#) ()
Construct ready to parse the request method.
- void [reset](#) ()

Reset to initial parser state.

- `template<typename InputIterator >`
`boost::tuple< boost::tribool,`
`InputIterator > parse (Request &req, InputIterator begin, InputIterator end)`

24.72.1 Detailed Description

Parser for incoming requests.

Definition at line 17 of file [RequestParser.hpp](#).

24.72.2 Constructor & Destructor Documentation

24.72.2.1 AIRINV::RequestParser::RequestParser ()

Construct ready to parse the request method.

Definition at line 13 of file [RequestParser.cpp](#).

24.72.3 Member Function Documentation

24.72.3.1 void AIRINV::RequestParser::reset ()

Reset to initial parser state.

Definition at line 18 of file [RequestParser.cpp](#).

24.72.3.2 `template<typename InputIterator > boost::tuple<boost::tribool, InputIterator> AIRINV::RequestParser::parse (Request & req, InputIterator begin, InputIterator end)` `[inline]`

Parse some data. The tribool return value is true when a complete request has been parsed, false if the data is invalid, indeterminate when more data is required. The InputIterator return value indicates how much of the input has been consumed.

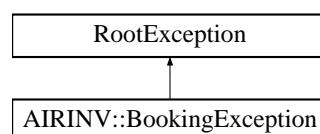
Definition at line 30 of file [RequestParser.hpp](#).

The documentation for this class was generated from the following files:

- [airinv/server/RequestParser.hpp](#)
- [airinv/server/RequestParser.cpp](#)

24.73 RootException Class Reference

Inheritance diagram for RootException:



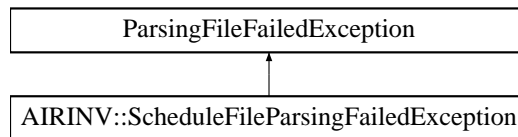
The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.74 AIRINV::ScheduleFileParsingFailedException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::ScheduleFileParsingFailedException:



Public Member Functions

- [ScheduleFileParsingFailedException](#) (const std::string &iWhat)

24.74.1 Detailed Description

The schedule input file can not be parsed.

Definition at line 40 of file [AIRINV_Types.hpp](#).

24.74.2 Constructor & Destructor Documentation

24.74.2.1 AIRINV::ScheduleFileParsingFailedException::ScheduleFileParsingFailedException (const std::string & iWhat) [inline]

Constructor.

Definition at line 46 of file [AIRINV_Types.hpp](#).

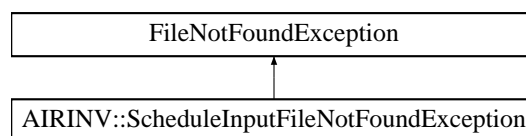
The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.75 AIRINV::ScheduleInputFileNotFoundedException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::ScheduleInputFileNotFoundedException:



Public Member Functions

- [ScheduleInputFileNotFoundedException](#) (const std::string &iWhat)

24.75.1 Detailed Description

The schedule input file can not be found or opened.

Definition at line 78 of file [AIRINV_Types.hpp](#).

24.75.2 Constructor & Destructor Documentation

24.75.2.1 AIRINV::ScheduleInputFileNotFoundException::ScheduleInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 83 of file [AIRINV_Types.hpp](#).

The documentation for this class was generated from the following file:

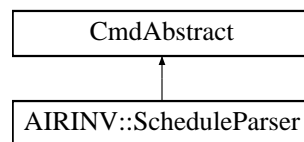
- [airinv/AIRINV_Types.hpp](#)

24.76 AIRINV::ScheduleParser Class Reference

Class wrapping the parser entry point.

```
#include <airinv/command/ScheduleParser.hpp>
```

Inheritance diagram for AIRINV::ScheduleParser:



Static Public Member Functions

- static void [generateInventories](#) (const stdair::Filename_T & iScheduleFilename, stdair::BomRoot &)

24.76.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 21 of file [ScheduleParser.hpp](#).

24.76.2 Member Function Documentation

24.76.2.1 void AIRINV::ScheduleParser::generateInventories (const stdair::Filename_T & iScheduleFilename, stdair::BomRoot & ioBomRoot) [static]

Parse the CSV file describing the airline schedules for the simulator, and generates the inventories accordingly.

Parameters

<i>const</i>	stdair::Filename_T& The file-name of the CSV-formatted schedule input file.
<i>stdair::Bom-Root&</i>	Root of the BOM tree.

Definition at line 20 of file [ScheduleParser.cpp](#).

References [AIRINV::InventoryManager::buildSimilarSegmentCabinSets\(\)](#), [AIRINV::InventoryManager::createDirectAccesses\(\)](#), [AIRINV::FlightPeriodFileParser::generateInventories\(\)](#), and [AIRINV::InventoryManager::setDefaultBidPriceVector\(\)](#).

Referenced by [AIRINV::AIRINV_Service::parseAndLoad\(\)](#).

The documentation for this class was generated from the following files:

- [airinv/command/ScheduleParser.hpp](#)
- [airinv/command/ScheduleParser.cpp](#)

24.77 AIRINV::SegmentCabinHelper Class Reference

Class representing the actual business functions for an airline segment-cabin.

```
#include <airinv/bom/SegmentCabinHelper.hpp>
```

Static Public Member Functions

- static void [updateFromReservation](#) (const stdair::FlightDate &, stdair::SegmentCabin &, const stdair::PartySize_T &)
- static void [buildPseudoBidPriceVector](#) (stdair::SegmentCabin &)
- static void [updateBookingControlsUsingPseudoBidPriceVector](#) (const stdair::SegmentCabin &)
- static void [updateAUs](#) (const stdair::SegmentCabin &)
- static void [updateAvailabilities](#) (const stdair::SegmentCabin &)
- static void [initialiseAU](#) (stdair::SegmentCabin &)

24.77.1 Detailed Description

Class representing the actual business functions for an airline segment-cabin.

Definition at line 23 of file [SegmentCabinHelper.hpp](#).

24.77.2 Member Function Documentation

24.77.2.1 void [AIRINV::SegmentCabinHelper::updateFromReservation](#) (const stdair::FlightDate & *iFlightDate*, stdair::SegmentCabin & *ioSegmentCabin*, const stdair::PartySize_T & *iNbOfBookings*) [static]

Update the segment-cabin with the reservation.

Definition at line 57 of file [SegmentCabinHelper.cpp](#).

References [AIRINV::FlightDateHelper::updateAvailabilityPool\(\)](#).

Referenced by [AIRINV::InventoryHelper::cancel\(\)](#), and [AIRINV::InventoryHelper::sell\(\)](#).

24.77.2.2 void [AIRINV::SegmentCabinHelper::buildPseudoBidPriceVector](#) (stdair::SegmentCabin & *ioSegmentCabin*) [static]

Build the pseudo bid price vector from the vectors of the leg-cabins.

Definition at line 82 of file [SegmentCabinHelper.cpp](#).

Referenced by [AIRINV::FlightDateHelper::updateBookingControls\(\)](#).

24.77.2.3 void [AIRINV::SegmentCabinHelper::updateBookingControlsUsingPseudoBidPriceVector](#) (const stdair::SegmentCabin & *iSegmentCabin*) [static]

Update the booking controls using the pseudo bid price vector.

Definition at line 126 of file [SegmentCabinHelper.cpp](#).

References [updateAUs\(\)](#).

Referenced by [AIRINV::FlightDateHelper::updateBookingControls\(\)](#).

24.77.2.4 void AIRINV::SegmentCabinHelper::updateAUs (const stdair::SegmentCabin & iSegmentCabin)
[static]

Update the authorisation levels using the booking limits.

Definition at line 158 of file [SegmentCabinHelper.cpp](#).

Referenced by [updateBookingControlsUsingPseudoBidPriceVector\(\)](#).

24.77.2.5 void AIRINV::SegmentCabinHelper::updateAvailabilities (const stdair::SegmentCabin & iSegmentCabin)
[static]

Update the availability of the booking classes.

Definition at line 190 of file [SegmentCabinHelper.cpp](#).

Referenced by [AIRINV::InventoryHelper::calculateAvailability\(\)](#), and [AIRINV::GuillotineBlockHelper::take-Snapshots\(\)](#).

24.77.2.6 void AIRINV::SegmentCabinHelper::initialiseAU (stdair::SegmentCabin & iSegmentCabin) [static]

Initialise the AU for the booking classes.

Definition at line 21 of file [SegmentCabinHelper.cpp](#).

Referenced by [AIRINV::SegmentDateHelper::fillFromRouting\(\)](#).

The documentation for this class was generated from the following files:

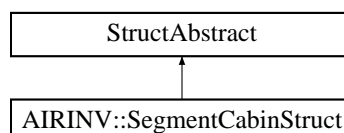
- [airinv/bom/SegmentCabinHelper.hpp](#)
- [airinv/bom/SegmentCabinHelper.cpp](#)

24.78 AIRINV::SegmentCabinStruct Struct Reference

Utility Structure for the parsing of SegmentCabin details.

`#include <airinv/bom/SegmentCabinStruct.hpp>`

Inheritance diagram for AIRINV::SegmentCabinStruct:



Public Member Functions

- void [fill](#) (stdair::SegmentCabin &) const
- const std::string [describe](#) () const

Public Attributes

- stdair::CabinCode_T [_cabinCode](#)
- stdair::NbOfBookings_T [_nbOfBookings](#)
- [FareFamilyStruct](#) [_itFareFamily](#)
- [FareFamilyStructList](#) [_fareFamilies](#)

24.78.1 Detailed Description

Utility Structure for the parsing of SegmentCabin details.

Definition at line 26 of file [SegmentCabinStruct.hpp](#).

24.78.2 Member Function Documentation

24.78.2.1 void AIRINV::SegmentCabinStruct::fill (stdair::SegmentCabin & ioSegmentCabin) const

Fill the SegmentCabin objects with the attributes of the [SegmentCabinStruct](#).

Definition at line 33 of file [SegmentCabinStruct.cpp](#).

24.78.2.2 const std::string AIRINV::SegmentCabinStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 15 of file [SegmentCabinStruct.cpp](#).

References [_cabinCode](#), [_fareFamilies](#), [_nbOfBookings](#), and [AIRINV::FareFamilyStruct::describe\(\)](#).

Referenced by [AIRINV::SegmentStruct::describe\(\)](#).

24.78.3 Member Data Documentation

24.78.3.1 stdair::CabinCode_T AIRINV::SegmentCabinStruct::_cabinCode

Definition at line 28 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#), [AIRINV::FlightDateStruct::addFareFamily\(\)](#), [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#).

24.78.3.2 stdair::NbOfBookings_T AIRINV::SegmentCabinStruct::_nbOfBookings

Definition at line 29 of file [SegmentCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#).

24.78.3.3 FareFamilyStruct AIRINV::SegmentCabinStruct::_itFareFamily

Definition at line 30 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#).

24.78.3.4 FareFamilyStructList_T AIRINV::SegmentCabinStruct::_fareFamilies

Definition at line 31 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#), [AIRINV::FlightDateStruct::addFareFamily\(\)](#), [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/bom/SegmentCabinStruct.hpp](#)

- [airinv/bom/SegmentCabinStruct.cpp](#)

24.79 AIRINV::SegmentDateHelper Class Reference

```
#include <airinv/bom/SegmentDateHelper.hpp>
```

Static Public Member Functions

- static void [fillFromRouting](#) (stdair::SegmentDate &)
- static void [updateElapsedTimeFromRouting](#) (stdair::SegmentDate &)
- static void [updateDistanceFromElapsedTime](#) (stdair::SegmentDate &)

24.79.1 Detailed Description

Class representing the actual business functions for an airline segment-date.

Definition at line 16 of file [SegmentDateHelper.hpp](#).

24.79.2 Member Function Documentation

24.79.2.1 void AIRINV::SegmentDateHelper::fillFromRouting (stdair::SegmentDate & *ioSegmentDate*)
[static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 18 of file [SegmentDateHelper.cpp](#).

References [AIRINV::SegmentCabinHelper::initialiseAU\(\)](#), and [updateElapsedTimeFromRouting\(\)](#).

24.79.2.2 void AIRINV::SegmentDateHelper::updateElapsedTimeFromRouting (stdair::SegmentDate & *ioSegmentDate*) [static]

Calculate and set the elapsed time according to the leg routing.

Actually, the elapsed time of the segment is the sum of the elapsed times of the routing legs, plus the stop-over times. The stop-over time is the difference between the board time of a routing leg, and the off time of the previous leg. That is, it is the time spent at the corresponding airport.

Of course, in case of mono-leg segments, there is no stop-over, and the elapsed time of the segment is equal to the elapsed time of the single routing leg.

Definition at line 73 of file [SegmentDateHelper.cpp](#).

References [updateDistanceFromElapsedTime\(\)](#).

Referenced by [fillFromRouting\(\)](#).

24.79.2.3 void AIRINV::SegmentDateHelper::updateDistanceFromElapsedTime (stdair::SegmentDate & *ioSegmentDate*) [static]

Method computing the distance of the segment (in kilometers).

Definition at line 116 of file [SegmentDateHelper.cpp](#).

Referenced by [updateElapsedTimeFromRouting\(\)](#).

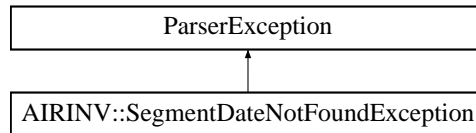
The documentation for this class was generated from the following files:

- [airinv/bom/SegmentDateHelper.hpp](#)
- [airinv/bom/SegmentDateHelper.cpp](#)

24.80 AIRINV::SegmentDateNotFoundException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::SegmentDateNotFoundException:



Public Member Functions

- [SegmentDateNotFoundException](#) (const std::string &iWhat)

24.80.1 Detailed Description

Specific exception when some BOM objects can not be found within the inventory.

Definition at line 54 of file [AIRINV_Types.hpp](#).

24.80.2 Constructor & Destructor Documentation

24.80.2.1 AIRINV::SegmentDateNotFoundException::SegmentDateNotFoundException (const std::string &iWhat) [inline]

Constructor.

Definition at line 59 of file [AIRINV_Types.hpp](#).

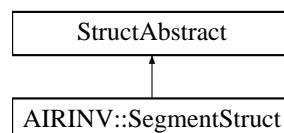
The documentation for this class was generated from the following file:

- [airinv/AIRINV_Types.hpp](#)

24.81 AIRINV::SegmentStruct Struct Reference

```
#include <airinv/bom/SegmentStruct.hpp>
```

Inheritance diagram for AIRINV::SegmentStruct:



Public Member Functions

- void [fill](#) (stdair::SegmentDate &) const
- const std::string [describe](#) () const

Public Attributes

- [stdair::AirportCode_T _boardingPoint](#)
- [stdair::AirportCode_T _offPoint](#)
- [stdair::Date_T _boardingDate](#)
- [stdair::Duration_T _boardingTime](#)
- [stdair::Date_T _offDate](#)
- [stdair::Duration_T _offTime](#)
- [stdair::Duration_T _elapsed](#)
- [SegmentCabinStructList_T _cabinList](#)

24.81.1 Detailed Description

Utility Structure for the parsing of Segment structures.

Definition at line 23 of file [SegmentStruct.hpp](#).

24.81.2 Member Function Documentation

24.81.2.1 void AIRINV::SegmentStruct::fill (stdair::SegmentDate & ioSegmentDate) const

Fill the SegmentDate objects with the attributes of the [SegmentStruct](#).

Definition at line 36 of file [SegmentStruct.cpp](#).

References [_boardingTime](#), [_elapsed](#), [_offDate](#), and [_offTime](#).

24.81.2.2 const std::string AIRINV::SegmentStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 14 of file [SegmentStruct.cpp](#).

References [_boardingPoint](#), [_boardingTime](#), [_cabinList](#), [_elapsed](#), [_offPoint](#), [_offTime](#), and [AIRINV::SegmentCabinStruct::describe\(\)](#).

Referenced by [AIRINV::FlightPeriodStruct::describe\(\)](#), and [AIRINV::FlightDateStruct::describe\(\)](#).

24.81.3 Member Data Documentation

24.81.3.1 stdair::AirportCode_T AIRINV::SegmentStruct::_boardingPoint

Definition at line 25 of file [SegmentStruct.hpp](#).

Referenced by [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#), [AIRINV::FlightDateStruct::addFareFamily\(\)](#), [AIRINV::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRINV::FlightDateStruct::addSegmentCabin\(\)](#), [AIRINV::FlightPeriodStruct::buildSegments\(\)](#), [AIRINV::FlightDateStruct::buildSegments\(\)](#), [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

24.81.3.2 stdair::AirportCode_T AIRINV::SegmentStruct::_offPoint

Definition at line 26 of file [SegmentStruct.hpp](#).

Referenced by [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#), [AIRINV::FlightDateStruct::addFareFamily\(\)](#), [AIRINV::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRINV::FlightDateStruct::addSegmentCabin\(\)](#), [AIRINV::FlightPeriodStruct::buildSegments\(\)](#), [AIRINV::FlightDateStruct::buildSegments\(\)](#), [describe\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#).

24.81.3.3 `stdair::Date_T AIRINV::SegmentStruct::_boardingDate`

Definition at line 27 of file [SegmentStruct.hpp](#).

24.81.3.4 `stdair::Duration_T AIRINV::SegmentStruct::_boardingTime`

Definition at line 28 of file [SegmentStruct.hpp](#).

Referenced by [describe\(\)](#), and [fill\(\)](#).

24.81.3.5 `stdair::Date_T AIRINV::SegmentStruct::_offDate`

Definition at line 29 of file [SegmentStruct.hpp](#).

Referenced by [fill\(\)](#).

24.81.3.6 `stdair::Duration_T AIRINV::SegmentStruct::_offTime`

Definition at line 30 of file [SegmentStruct.hpp](#).

Referenced by [describe\(\)](#), and [fill\(\)](#).

24.81.3.7 `stdair::Duration_T AIRINV::SegmentStruct::_elapsed`

Definition at line 31 of file [SegmentStruct.hpp](#).

Referenced by [describe\(\)](#), and [fill\(\)](#).

24.81.3.8 `SegmentCabinStructList_T AIRINV::SegmentStruct::_cabinList`

Definition at line 32 of file [SegmentStruct.hpp](#).

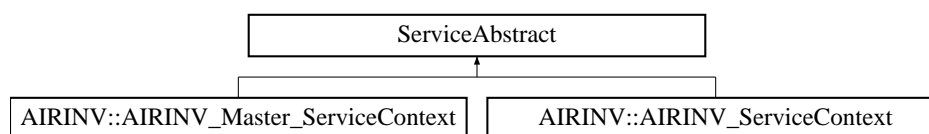
Referenced by [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#), [AIRINV::FlightDateStruct::addFareFamily\(\)](#), [AIRINV::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRINV::FlightDateStruct::addSegmentCabin\(\)](#), [describe\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/bom/SegmentStruct.hpp](#)
- [airinv/bom/SegmentStruct.cpp](#)

24.82 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



The documentation for this class was generated from the following file:

- [airinv/service/AIRINV_Master_ServiceContext.hpp](#)

24.83 AIRINV::ServiceAbstract Class Reference

```
#include <airinv/service/ServiceAbstract.hpp>
```

Public Member Functions

- virtual [~ServiceAbstract](#) ()
- virtual void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Protected Member Functions

- [ServiceAbstract](#) ()

24.83.1 Detailed Description

Base class for the Service layer.

Definition at line 14 of file [ServiceAbstract.hpp](#).

24.83.2 Constructor & Destructor Documentation

24.83.2.1 virtual **AIRINV::ServiceAbstract::~~ServiceAbstract** () [inline, virtual]

Destructor.

Definition at line 18 of file [ServiceAbstract.hpp](#).

24.83.2.2 **AIRINV::ServiceAbstract::ServiceAbstract** () [inline, protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 30 of file [ServiceAbstract.hpp](#).

24.83.3 Member Function Documentation

24.83.3.1 virtual void **AIRINV::ServiceAbstract::toStream** (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 22 of file [ServiceAbstract.hpp](#).

24.83.3.2 virtual void **AIRINV::ServiceAbstract::fromStream** (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Definition at line 26 of file [ServiceAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this class was generated from the following file:

- airinv/service/[ServiceAbstract.hpp](#)

24.84 swift::SKeymap Class Reference

The readline keymap wrapper.

```
#include <airinv/ui/cmdline/SReadline.hpp>
```

Public Member Functions

- [SKeymap](#) (bool PrintableBound=false)
Creates a new keymap.
- [SKeymap](#) (Keymap Pattern)
Creates a new keymap which is a copy of Pattern.
- [~SKeymap](#) ()
Frees the allocated keymap.
- void [Bind](#) (int Key, KeyCallback Callback)
Binds the given key to a function.
- void [Unbind](#) (int Key)
Unbinds the given key.
- [SKeymap](#) (const [SKeymap](#) &rhs)
Copy constructor.
- [SKeymap](#) & [operator=](#) (const [SKeymap](#) &rhs)
operator=

Friends

- class [SReadline](#)

24.84.1 Detailed Description

The readline keymap wrapper.

Attention: It is not thread safe! Supports: key binding, key unbinding

Definition at line 307 of file [SReadline.hpp](#).

24.84.2 Constructor & Destructor Documentation

24.84.2.1 swift::SKeymap::SKeymap (bool PrintableBound = false) [inline, explicit]

Creates a new keymap.

Parameters

<i>PrintableBound</i>	if true - the printable characters are bound if false - the keymap is empty
-----------------------	---

Definition at line 319 of file [SReadline.hpp](#).

24.84.2.2 swift::SKeymap::SKeymap (Keymap Pattern) [inline, explicit]

Creates a new keymap which is a copy of Pattern.

Parameters

<i>Pattern</i>	A keymap to be copied.
----------------	------------------------

Definition at line 342 of file [SReadline.hpp](#).

24.84.2.3 swift::SKeymap::~~SKeymap () `[inline]`

Frees the allocated keymap.

Definition at line 354 of file [SReadline.hpp](#).

24.84.2.4 swift::SKeymap::SKeymap (const SKeymap & rhs) `[inline]`

Copy constructor.

Parameters

<i>rhs</i>	Right hand side object of SKeymap
------------	---

Definition at line 395 of file [SReadline.hpp](#).

24.84.3 Member Function Documentation**24.84.3.1 void swift::SKeymap::Bind (int Key, KeyCallback Callback)** `[inline]`

Binds the given key to a function.

Parameters

<i>Key</i>	A key to be bound
<i>Callback</i>	A function to be called when the Key is pressed

Definition at line 366 of file [SReadline.hpp](#).

24.84.3.2 void swift::SKeymap::Unbind (int Key) `[inline]`

Unbinds the given key.

Parameters

<i>Key</i>	A key to be unbound
------------	---------------------

Definition at line 381 of file [SReadline.hpp](#).

24.84.3.3 SKeymap& swift::SKeymap::operator= (const SKeymap & rhs) `[inline]`

operator=

Parameters

<i>rhs</i>	Right hand side object of SKeymap
------------	---

Definition at line 407 of file [SReadline.hpp](#).

24.84.4 Friends And Related Function Documentation**24.84.4.1 friend class SReadline** `[friend]`

Definition at line 415 of file [SReadline.hpp](#).

The documentation for this class was generated from the following file:

- [airinv/ui/cmdline/SReadline.hpp](#)

24.85 swift::SReadline Class Reference

The readline library wrapper.

```
#include <airinv/ui/cmdline/SReadline.hpp>
```

Public Member Functions

- [SReadline](#) (const size_t Limit=DefaultHistoryLimit)
Constructs the object, sets the completion function.
- [SReadline](#) (const std::string &historyFileName, const size_t Limit=DefaultHistoryLimit)
Constructs the object, sets the completion function, loads history.
- [~SReadline](#) ()
Saves the session history (if the file name was provided) and destroys the object.
- std::string [GetLine](#) (const std::string &Prompt)
Gets a single line from a user.
- template<typename Container >
std::string [GetLine](#) (const std::string &Prompt, Container &ReadTokens)
Gets a single line from a user.
- template<typename Container >
std::string [GetLine](#) (const std::string &Prompt, Container &ReadTokens, bool &BreakOut)
Gets a single line from a user.
- std::string [GetLine](#) (const std::string &Prompt, bool &BreakOut)
Gets a single line from a user.
- template<typename ContainerType >
void [GetHistory](#) (ContainerType &Container)
Fills the given container with the current history list.
- bool [SaveHistory](#) (std::ostream &OS)
Saves the history to the given file stream.
- bool [SaveHistory](#) (const std::string &FileName)
Saves the history to the given file.
- void [ClearHistory](#) ()
Clears the history. Does not affect the file where the previous session history is saved.
- bool [LoadHistory](#) (std::istream &IS)
Loads a history from a file stream.
- bool [LoadHistory](#) (const std::string &FileName)
Loads a history from the given file.
- template<typename ContainerType >
void [RegisterCompletions](#) (const ContainerType &Container)
Allows to register custom completers.
- void [SetKeymap](#) (SKeymap &NewKeymap)
Sets the given keymap.

24.85.1 Detailed Description

The readline library wrapper.

Attention: It is not thread safe! Supports: editing, history, custom completers

Definition at line [424](#) of file [SReadline.hpp](#).

24.85.2 Constructor & Destructor Documentation

24.85.2.1 **swift::SReadline::SReadline** (**const size_t** *Limit* = `DefaultHistoryLimit`) [inline]

Constructs the object, sets the completion function.

Parameters

<i>Limit</i>	History size
--------------	--------------

Definition at line 431 of file [SReadline.hpp](#).

24.85.2.2 **swift::SReadline::SReadline** (**const std::string &** *historyFileName*, **const size_t** *Limit* = `DefaultHistoryLimit`) [inline]

Constructs the object, sets the completion function, loads history.

Parameters

<i>historyFileName</i>	File name to load history from
<i>Limit</i>	History size

Definition at line 446 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

24.85.2.3 **swift::SReadline::~~SReadline** () [inline]

Saves the session history (if the file name was provided) and destroys the object.

Definition at line 460 of file [SReadline.hpp](#).

References [SaveHistory\(\)](#).

24.85.3 Member Function Documentation

24.85.3.1 **std::string swift::SReadline::GetLine** (**const std::string &** *Prompt*) [inline]

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
---------------	------------------

Returns

A string which was actually inputed

Definition at line 471 of file [SReadline.hpp](#).

Referenced by [GetLine\(\)](#).

24.85.3.2 **template<typename Container > std::string swift::SReadline::GetLine** (**const std::string &** *Prompt*, **Container &** *ReadTokens*) [inline]

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>ReadTokens</i>	A user inputed string splitted into tokens. The container is cleared first

Returns

A string which was actually inputed

Definition at line 485 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

24.85.3.3 `template<typename Container > std::string swift::SReadline::GetLine (const std::string & Prompt, Container & ReadTokens, bool & BreakOut) [inline]`

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found
<i>ReadTokens</i>	A user inputed string splitted into tokens. The container is cleared first

Returns

A string which was actually inputed

Definition at line 500 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

24.85.3.4 `std::string swift::SReadline::GetLine (const std::string & Prompt, bool & BreakOut) [inline]`

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found

Returns

A string which was actually inputed

Definition at line 515 of file [SReadline.hpp](#).

24.85.3.5 `template<typename ContainerType > void swift::SReadline::GetHistory (ContainerType & Container) [inline]`

Fills the given container with the current history list.

Does not clear the given container

Definition at line 550 of file [SReadline.hpp](#).

24.85.3.6 `bool swift::SReadline::SaveHistory (std::ostream & OS) [inline]`

Saves the history to the given file stream.

Parameters

<i>OS</i>	output file stream
-----------	--------------------

Returns

true if success

Definition at line 562 of file [SReadline.hpp](#).

Referenced by [SaveHistory\(\)](#), and [~SReadline\(\)](#).

24.85.3.7 `bool swift::SReadline::SaveHistory (const std::string & FileName)` `[inline]`

Saves the history to the given file.

Parameters

<i>FileName</i>	File name to save the history to
-----------------	----------------------------------

Returns

true if success

Definition at line 579 of file [SReadline.hpp](#).

References [SaveHistory\(\)](#).

24.85.3.8 `void swift::SReadline::ClearHistory ()` `[inline]`

Clears the history. Does not affect the file where the previous session history is saved.

Definition at line 592 of file [SReadline.hpp](#).

Referenced by [LoadHistory\(\)](#).

24.85.3.9 `bool swift::SReadline::LoadHistory (std::istream & IS)` `[inline]`

Loads a history from a file stream.

Parameters

<i>IS</i>	Input file stream
-----------	-------------------

Returns

true if success

Definition at line 602 of file [SReadline.hpp](#).

References [ClearHistory\(\)](#).

Referenced by [LoadHistory\(\)](#), and [SReadline\(\)](#).

24.85.3.10 `bool swift::SReadline::LoadHistory (const std::string & FileName)` `[inline]`

Loads a history from the given file.

Parameters

<i>FileName</i>	File name to be load from
-----------------	---------------------------

Returns

true if success

Definition at line 627 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

24.85.3.11 `template<typename ContainerType > void swift::SReadline::RegisterCompletions (const ContainerType & Container) [inline]`

Allows to register custom completers.

Supports a special keyword: file. It means to use the standard file name completer.

For example the given container elements could be as follows:

- command1 opt1
- command1 opt2 file
- command2
- command2 opt1

Each container element must describe a single possible command line. The container element must have a conversion to `std::string` operator.

Parameters

<i>Container</i>	A container which has all the user possible commands.
------------------	---

Definition at line [656](#) of file [SReadline.hpp](#).

24.85.3.12 `void swift::SReadline::SetKeymap (SKeymap & NewKeymap) [inline]`

Sets the given keymap.

Parameters

<i>NewKeymap</i>	The keymap that should be used from now.
------------------	--

Definition at line [673](#) of file [SReadline.hpp](#).

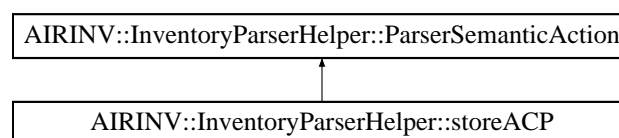
The documentation for this class was generated from the following file:

- [airinv/ui/cmdline/SReadline.hpp](#)

24.86 AIRINV::InventoryParserHelper::storeACP Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeACP:



Public Member Functions

- [storeACP](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.86.1 Detailed Description

Store the parsed Average Cancellation Percentage (ACP).

Definition at line 189 of file [InventoryParserHelper.hpp](#).

24.86.2 Constructor & Destructor Documentation

24.86.2.1 AIRINV::InventoryParserHelper::storeACP::storeACP ([FlightDateStruct](#) & [ioFlightDate](#))

Actor Constructor.

Definition at line 318 of file [InventoryParserHelper.cpp](#).

24.86.3 Member Function Documentation

24.86.3.1 void AIRINV::InventoryParserHelper::storeACP::operator() ([double iReal](#)) const

Actor Function (functor).

Definition at line 323 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_acp](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itLegCabin](#).

24.86.4 Member Data Documentation

24.86.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [A-](#)

[AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

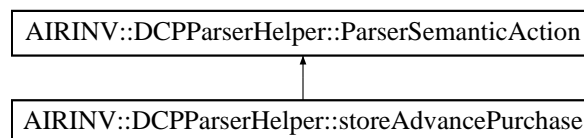
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.87 AIRINV::DCPParserHelper::storeAdvancePurchase Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeAdvancePurchase:



Public Member Functions

- [storeAdvancePurchase](#) (DCPRuleStruct &)
- void [operator\(\)](#) (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.87.1 Detailed Description

Store the parsed advance purchase days.

Definition at line 138 of file [DCPParserHelper.hpp](#).

24.87.2 Constructor & Destructor Documentation

24.87.2.1 AIRINV::DCPParserHelper::storeAdvancePurchase::storeAdvancePurchase (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 208 of file [DCPParserHelper.cpp](#).

24.87.3 Member Function Documentation

24.87.3.1 void AIRINV::DCPParserHelper::storeAdvancePurchase::operator() (unsigned int iAdvancePurchase, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 213 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.87.4 Member Data Documentation

24.87.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

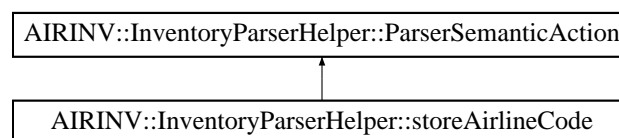
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.88 AIRINV::InventoryParserHelper::storeAirlineCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeAirlineCode:



Public Member Functions

- [storeAirlineCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.88.1 Detailed Description

Store the parsed airline code.

Definition at line 45 of file [InventoryParserHelper.hpp](#).

24.88.2 Constructor & Destructor Documentation

24.88.2.1 AIRINV::InventoryParserHelper::storeAirlineCode::storeAirlineCode (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 44 of file [InventoryParserHelper.cpp](#).

24.88.3 Member Function Documentation

24.88.3.1 void AIRINV::InventoryParserHelper::storeAirlineCode::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 49 of file [InventoryParserHelper.cpp](#).

References [AIRINV::FlightDateStruct::_airlineCode](#), [AIRINV::LegCabinStruct::_bucketList](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::BookingClassStruct::_classCode](#), [AIRINV::FareFamilyStruct::_classList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), [AIRINV::FlightDateStruct::_itBucket](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itLegCabin](#), [AIRINV::FlightDateStruct::_itSegment](#), [AIRINV::FlightDateStruct::_itSegmentCabin](#), [AIRINV::FlightDateStruct::_legList](#), [AIRINV::FlightDateStruct::_segmentList](#), and [AIRINV::BucketStruct::_yieldRangeUpperValue](#).

24.88.4 Member Data Documentation

24.88.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)\(\)](#), [AIRINV::InventoryParserHelper::](#)

[::storeSegmentAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#)(), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#)).

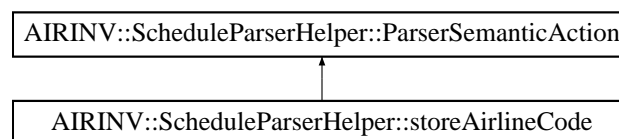
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.89 AIRINV::ScheduleParserHelper::storeAirlineCode Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeAirlineCode:



Public Member Functions

- [storeAirlineCode](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.89.1 Detailed Description

Store the parsed airline code.

Definition at line 37 of file [ScheduleParserHelper.hpp](#).

24.89.2 Constructor & Destructor Documentation

24.89.2.1 AIRINV::ScheduleParserHelper::storeAirlineCode::storeAirlineCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 33 of file [ScheduleParserHelper.cpp](#).

24.89.3 Member Function Documentation

24.89.3.1 void AIRINV::ScheduleParserHelper::storeAirlineCode::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 38 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_airlineCode](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRINV::FlightPeriodStruct::_legList](#).

24.89.4 Member Data Documentation

24.89.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

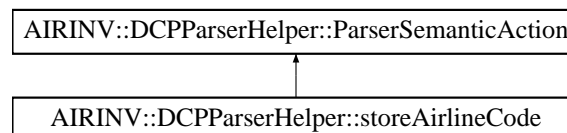
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.90 AIRINV::DCPParserHelper::storeAirlineCode Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeAirlineCode:



Public Member Functions

- [storeAirlineCode](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.90.1 Detailed Description

Store the parsed airline code.

Definition at line 198 of file [DCPParserHelper.hpp](#).

24.90.2 Constructor & Destructor Documentation

24.90.2.1 AIRINV::DCPParserHelper::storeAirlineCode::storeAirlineCode (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 329 of file [DCPParserHelper.cpp](#).

24.90.3 Member Function Documentation

24.90.3.1 void AIRINV::DCPParserHelper::storeAirlineCode::operator() (std::vector< char > iChar, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 334 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.90.4 Member Data Documentation

24.90.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

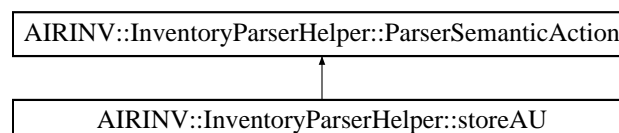
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.91 AIRINV::InventoryParserHelper::storeAU Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeAU:



Public Member Functions

- [storeAU](#) (FlightDateStruct &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.91.1 Detailed Description

Store the parsed Authorisation Level (AU).

Definition at line 149 of file [InventoryParserHelper.hpp](#).

24.91.2 Constructor & Destructor Documentation

24.91.2.1 AIRINV::InventoryParserHelper::storeAU::storeAU ([FlightDateStruct](#) & [ioFlightDate](#))

Actor Constructor.

Definition at line 263 of file [InventoryParserHelper.cpp](#).

24.91.3 Member Function Documentation

24.91.3.1 void AIRINV::InventoryParserHelper::storeAU::operator() ([double iReal](#)) const

Actor Function (functor).

Definition at line 268 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_au](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itLegCabin](#).

24.91.4 Member Data Documentation

24.91.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [A-](#)

[AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

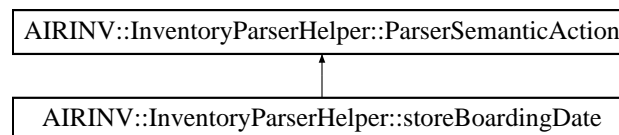
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.92 AIRINV::InventoryParserHelper::storeBoardingDate Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeBoardingDate:



Public Member Functions

- [storeBoardingDate](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.92.1 Detailed Description

Store the boarding date.

Definition at line 101 of file [InventoryParserHelper.hpp](#).

24.92.2 Constructor & Destructor Documentation

24.92.2.1 AIRINV::InventoryParserHelper::storeBoardingDate::storeBoardingDate ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 172 of file [InventoryParserHelper.cpp](#).

24.92.3 Member Function Documentation

24.92.3.1 void AIRINV::InventoryParserHelper::storeBoardingDate::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 177 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingDate](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLeg](#), and [AIRINV::FlightDateStruct::getDate\(\)](#).

24.92.4 Member Data Documentation

24.92.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailality::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

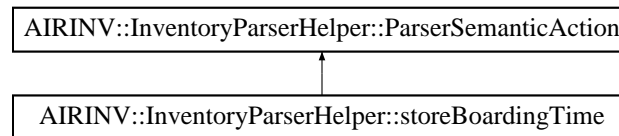
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.93 AIRINV::InventoryParserHelper::storeBoardingTime Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeBoardingTime:



Public Member Functions

- [storeBoardingTime](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.93.1 Detailed Description

Store the boarding time.

Definition at line 109 of file [InventoryParserHelper.hpp](#).

24.93.2 Constructor & Destructor Documentation

24.93.2.1 AIRINV::InventoryParserHelper::storeBoardingTime::storeBoardingTime ([FlightDateStruct](#) & [ioFlightDate](#))

Actor Constructor.

Definition at line 183 of file [InventoryParserHelper.cpp](#).

24.93.3 Member Function Documentation

24.93.3.1 void AIRINV::InventoryParserHelper::storeBoardingTime::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 188 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingTime](#), [AIRINV::FlightDateStruct::_dateOffSet](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itSeconds](#), and [AIRINV::FlightDateStruct::getTime\(\)](#).

24.93.4 Member Data Documentation

24.93.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::store-](#)

OffDate::operator(), AIRINV::InventoryParserHelper::storeOffTime::operator(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator(), AIRINV::InventoryParserHelper::storeAU::operator(), AIRINV::InventoryParserHelper::storeUPR::operator(), AIRINV::InventoryParserHelper::storeBookingCounter::operator(), AIRINV::InventoryParserHelper::storeNAV::operator(), AIRINV::InventoryParserHelper::storeGAV::operator(), AIRINV::InventoryParserHelper::storeACP::operator(), AIRINV::InventoryParserHelper::storeETB::operator(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator(), AIRINV::InventoryParserHelper::storeBucketAvailality::operator(), AIRINV::InventoryParserHelper::storeSeatIndex::operator(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator(), AIRINV::InventoryParserHelper::storeClassCode::operator(), AIRINV::InventoryParserHelper::storeSubclassCode::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

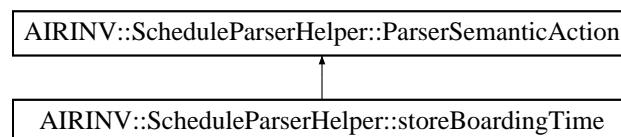
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.94 AIRINV::ScheduleParserHelper::storeBoardingTime Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeBoardingTime:



Public Member Functions

- [storeBoardingTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.94.1 Detailed Description

Store the boarding time.

Definition at line 93 of file [ScheduleParserHelper.hpp](#).

24.94.2 Constructor & Destructor Documentation

24.94.2.1 AIRINV::ScheduleParserHelper::storeBoardingTime::storeBoardingTime (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 156 of file [ScheduleParserHelper.cpp](#).

24.94.3 Member Function Documentation

24.94.3.1 void AIRINV::ScheduleParserHelper::storeBoardingTime::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 161 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingTime](#), [AIRINV::FlightPeriodStruct::_dateOffset](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_itSeconds](#), and [AIRINV::FlightPeriodStruct::getTime\(\)](#).

24.94.4 Member Data Documentation

24.94.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

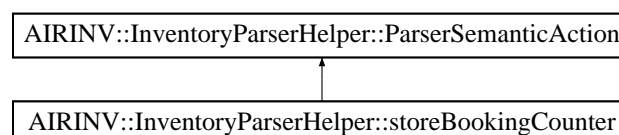
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.95 AIRINV::InventoryParserHelper::storeBookingCounter Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeBookingCounter:



Public Member Functions

- [storeBookingCounter](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.95.1 Detailed Description

Store the parsed booking counter.

Definition at line 165 of file [InventoryParserHelper.hpp](#).

24.95.2 Constructor & Destructor Documentation

24.95.2.1 AIRINV::InventoryParserHelper::storeBookingCounter::storeBookingCounter ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 285 of file [InventoryParserHelper.cpp](#).

24.95.3 Member Function Documentation

24.95.3.1 void AIRINV::InventoryParserHelper::storeBookingCounter::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 290 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::LegCabinStruct::_nbOfBookings](#).

24.95.4 Member Data Documentation

24.95.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoarding](#)

Point::operator>(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator>(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator>(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator>(), AIRINV::InventoryParserHelper::storeClassCode::operator>(), AIRINV::InventoryParserHelper::storeSubclassCode::operator>(), AIRINV::InventoryParserHelper::storeParentClassCode::operator>(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator>(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator>(), AIRINV::InventoryParserHelper::storeProtection::operator>(), AIRINV::InventoryParserHelper::storeNego::operator>(), AIRINV::InventoryParserHelper::storeNoShow::operator>(), AIRINV::InventoryParserHelper::storeOverbooking::operator>(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator>(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator>(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator>(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator>(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator>(), AIRINV::InventoryParserHelper::storeClassETB::operator>(), AIRINV::InventoryParserHelper::storeClassAvailability::operator>(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator>(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator>(), AIRINV::InventoryParserHelper::storeFamilyCode::operator>(), AIRINV::InventoryParserHelper::storeFClasses::operator>(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()).

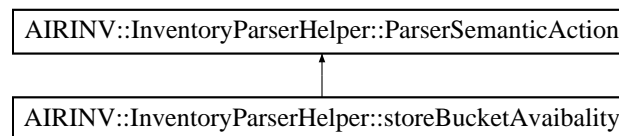
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.96 AIRINV::InventoryParserHelper::storeBucketAvaibility Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeBucketAvaibility:



Public Member Functions

- [storeBucketAvaibility](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.96.1 Detailed Description

Store the parsed bucket availability.

Definition at line 213 of file [InventoryParserHelper.hpp](#).

24.96.2 Constructor & Destructor Documentation

24.96.2.1 AIRINV::InventoryParserHelper::storeBucketAvaibility::storeBucketAvaibility ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 360 of file [InventoryParserHelper.cpp](#).

24.96.3 Member Function Documentation

24.96.3.1 void AIRINV::InventoryParserHelper::storeBucketAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 365 of file [InventoryParserHelper.cpp](#).

References [AIRINV::BucketStruct::_availability](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itBucket](#).

24.96.4 Member Data Documentation

24.96.4.1 **FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate**
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

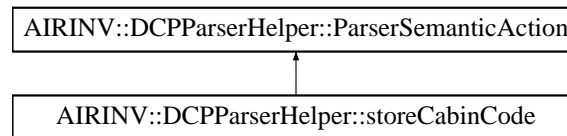
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.97 AIRINV::DCPPParserHelper::storeCabinCode Struct Reference

```
#include <airinv/command/vault/DCPPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPPParserHelper::storeCabinCode:



Public Member Functions

- [storeCabinCode](#) (DCPRuleStruct &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.97.1 Detailed Description

Store the cabin code.

Definition at line 118 of file [DCPParserHelper.hpp](#).

24.97.2 Constructor & Destructor Documentation

24.97.2.1 AIRINV::DCPParserHelper::storeCabinCode::storeCabinCode (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 166 of file [DCPParserHelper.cpp](#).

24.97.3 Member Function Documentation

24.97.3.1 void AIRINV::DCPParserHelper::storeCabinCode::operator() (char *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 171 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.97.4 Member Data Documentation

24.97.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::](#)

[::storeDCP::operator\(\)](#)(), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#)(), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#)(), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#)).

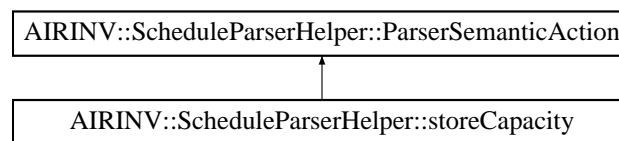
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.98 AIRINV::ScheduleParserHelper::storeCapacity Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeCapacity:



Public Member Functions

- [storeCapacity](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.98.1 Detailed Description

Store the parsed capacity.

Definition at line 125 of file [ScheduleParserHelper.hpp](#).

24.98.2 Constructor & Destructor Documentation

24.98.2.1 AIRINV::ScheduleParserHelper::storeCapacity::storeCapacity ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 228 of file [ScheduleParserHelper.cpp](#).

24.98.3 Member Function Documentation

24.98.3.1 void AIRINV::ScheduleParserHelper::storeCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 233 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::LegStruct::_cabinList](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_itLegCabin](#), and [AIRINV::LegCabinStruct::_saleableCapacity](#).

24.98.4 Member Data Documentation

24.98.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

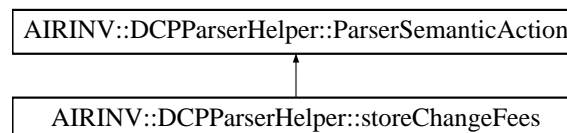
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.99 AIRINV::DCPPParserHelper::storeChangeFees Struct Reference

```
#include <airinv/command/vault/DCPPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPPParserHelper::storeChangeFees:



Public Member Functions

- [storeChangeFees](#) (DCPRuleStruct &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.99.1 Detailed Description

Store the parsed change fees.

Definition at line 158 of file [DCPPParserHelper.hpp](#).

24.99.2 Constructor & Destructor Documentation

24.99.2.1 AIRINV::DCPParserHelper::storeChangeFees::storeChangeFees (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 248 of file [DCPParserHelper.cpp](#).

24.99.3 Member Function Documentation

24.99.3.1 void AIRINV::DCPParserHelper::storeChangeFees::operator() (char iChangefees, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 253 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.99.4 Member Data Documentation

24.99.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

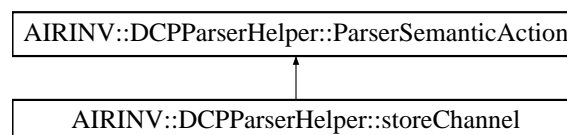
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.100 AIRINV::DCPParserHelper::storeChannel Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeChannel:



Public Member Functions

- [storeChannel](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [DCPRuleStruct & _DCPRule](#)

24.100.1 Detailed Description

Store the channel distribution.

Definition at line 128 of file [DCPParserHelper.hpp](#).

24.100.2 Constructor & Destructor Documentation

24.100.2.1 AIRINV::DCPParserHelper::storeChannel::storeChannel (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 187 of file [DCPParserHelper.cpp](#).

24.100.3 Member Function Documentation

24.100.3.1 void AIRINV::DCPParserHelper::storeChannel::operator() (std::vector< char > iChar, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 192 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.100.4 Member Data Documentation

24.100.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

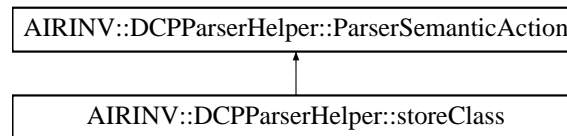
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.101 AIRINV::DCPParserHelper::storeClass Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeClass:



Public Member Functions

- [storeClass](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.101.1 Detailed Description

Store the parsed class.

Definition at line 208 of file [DCPParserHelper.hpp](#).

24.101.2 Constructor & Destructor Documentation

24.101.2.1 AIRINV::DCPParserHelper::storeClass::storeClass (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 376 of file [DCPParserHelper.cpp](#).

24.101.3 Member Function Documentation

24.101.3.1 void AIRINV::DCPParserHelper::storeClass::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 381 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.101.4 Member Data Documentation

24.101.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#),

[AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

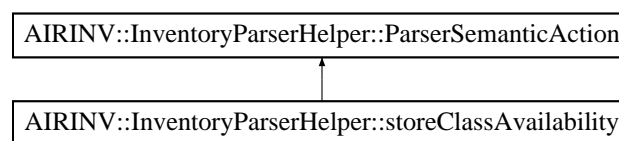
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.102 AIRINV::InventoryParserHelper::storeClassAvailability Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassAvailability:



Public Member Functions

- [storeClassAvailability](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.102.1 Detailed Description

Store the parsed number of net class availability (at booking class level).

Definition at line 383 of file [InventoryParserHelper.hpp](#).

24.102.2 Constructor & Destructor Documentation

24.102.2.1 AIRINV::InventoryParserHelper::storeClassAvailability::storeClassAvailability ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 670 of file [InventoryParserHelper.cpp](#).

24.102.3 Member Function Documentation

24.102.3.1 void AIRINV::InventoryParserHelper::storeClassAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 675 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_it-BookingClass](#), and [AIRINV::BookingClassStruct::_netClassAvailability](#).

24.102.4 Member Data Documentation

24.102.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

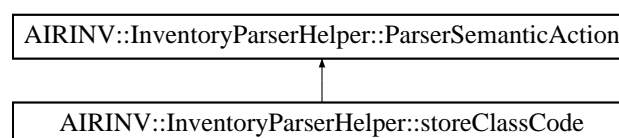
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.103 AIRINV::InventoryParserHelper::storeClassCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassCode:



Public Member Functions

- [storeClassCode](#) ([FlightDateStruct](#) &)

- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.103.1 Detailed Description

Store the parsed booking class code.

Definition at line 261 of file [InventoryParserHelper.hpp](#).

24.103.2 Constructor & Destructor Documentation

24.103.2.1 AIRINV::InventoryParserHelper::storeClassCode::storeClassCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 492 of file [InventoryParserHelper.cpp](#).

24.103.3 Member Function Documentation

24.103.3.1 void AIRINV::InventoryParserHelper::storeClassCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 497 of file [InventoryParserHelper.cpp](#).

References [AIRINV::BookingClassStruct::_classCode](#), [AIRINV::FareFamilyStruct::_classList](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), and [AIRINV::FlightDateStruct::_itSegmentCabin](#).

24.103.4 Member Data Documentation

24.103.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::-](#)

[InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

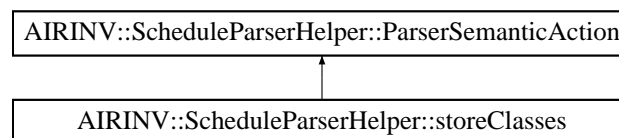
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.104 AIRINV::ScheduleParserHelper::storeClasses Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeClasses:



Public Member Functions

- [storeClasses](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.104.1 Detailed Description

Store the parsed list of class codes.

Definition at line 168 of file [ScheduleParserHelper.hpp](#).

24.104.2 Constructor & Destructor Documentation

24.104.2.1 AIRINV::ScheduleParserHelper::storeClasses::storeClasses ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 310 of file [ScheduleParserHelper.cpp](#).

24.104.3 Member Function Documentation

24.104.3.1 void AIRINV::ScheduleParserHelper::storeClasses::operator()(iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 315 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRINV::FareFamilyStruct::_classes](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightPeriodStruct::_itSegment](#), [AIRINV::FlightPeriodStruct::_itSegmentCabin](#), and [AIRINV::FlightPeriodStruct::addSegmentCabin\(\)](#).

24.104.4 Member Data Documentation

24.104.4.1 **FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod**
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

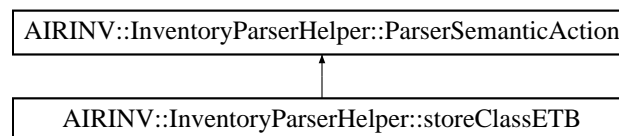
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.105 AIRINV::InventoryParserHelper::storeClassETB Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassETB:



Public Member Functions

- [storeClassETB](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.105.1 Detailed Description

Store the parsed expected to board number (at booking class level).

Definition at line 374 of file [InventoryParserHelper.hpp](#).

24.105.2 Constructor & Destructor Documentation

24.105.2.1 AIRINV::InventoryParserHelper::storeClassETB::storeClassETB ([FlightDateStruct](#) & [ioFlightDate](#))

Actor Constructor.

Definition at line 658 of file [InventoryParserHelper.cpp](#).

24.105.3 Member Function Documentation

24.105.3.1 void AIRINV::InventoryParserHelper::storeClassETB::operator() ([double iReal](#)) const

Actor Function (functor).

Definition at line 663 of file [InventoryParserHelper.cpp](#).

References [AIRINV::BookingClassStruct::_etb](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itBookingClass](#).

24.105.4 Member Data Documentation

24.105.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailablity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#),

AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFCClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

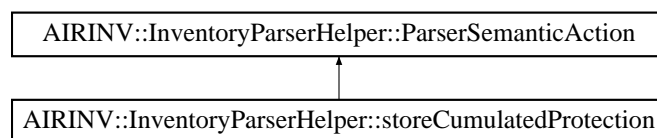
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.106 AIRINV::InventoryParserHelper::storeCumulatedProtection Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeCumulatedProtection:



Public Member Functions

- [storeCumulatedProtection](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.106.1 Detailed Description

Store the parsed cumulated protection (at booking class level).

Definition at line 293 of file [InventoryParserHelper.hpp](#).

24.106.2 Constructor & Destructor Documentation

24.106.2.1 AIRINV::InventoryParserHelper::storeCumulatedProtection::storeCumulatedProtection ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 547 of file [InventoryParserHelper.cpp](#).

24.106.3 Member Function Documentation

24.106.3.1 void AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(double *iReal*) const

Actor Function (functor).

Definition at line 552 of file [InventoryParserHelper.cpp](#).

References [AIRINV::BookingClassStruct::_cumulatedProtection](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itBookingClass](#).

24.106.4 Member Data Documentation

24.106.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

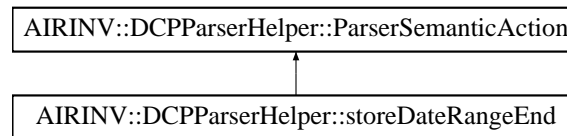
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.107 AIRINV::DCPParserHelper::storeDateRangeEnd Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDateRangeEnd:



Public Member Functions

- [storeDateRangeEnd](#) (DCPRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.107.1 Detailed Description

Store the parsed end of the date range.

Definition at line 78 of file [DCPParserHelper.hpp](#).

24.107.2 Constructor & Destructor Documentation

24.107.2.1 AIRINV::DCPParserHelper::storeDateRangeEnd::storeDateRangeEnd (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 101 of file [DCPParserHelper.cpp](#).

24.107.3 Member Function Documentation

24.107.3.1 void AIRINV::DCPParserHelper::storeDateRangeEnd::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 106 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.107.4 Member Data Documentation

24.107.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#),

[AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

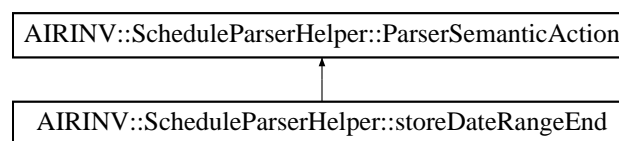
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.108 AIRINV::ScheduleParserHelper::storeDateRangeEnd Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDateRangeEnd:



Public Member Functions

- [storeDateRangeEnd](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.108.1 Detailed Description

Store the end of the date range.

Definition at line 61 of file [ScheduleParserHelper.hpp](#).

24.108.2 Constructor & Destructor Documentation

24.108.2.1 AIRINV::ScheduleParserHelper::storeDateRangeEnd::storeDateRangeEnd ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 76 of file [ScheduleParserHelper.cpp](#).

24.108.3 Member Function Documentation

24.108.3.1 void AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 81 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_dateRange](#), [AIRINV::FlightPeriodStruct::_dateRangeEnd](#), [AIRINV::FlightPeriodStruct::_dateRangeStart](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itSeconds](#), and [AIRINV::FlightPeriodStruct::getDate\(\)](#).

24.108.4 Member Data Documentation

24.108.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

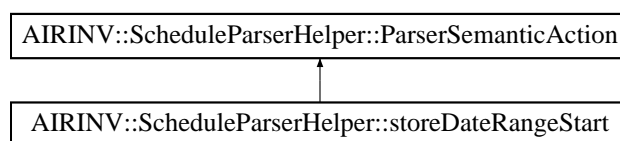
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.109 AIRINV::ScheduleParserHelper::storeDateRangeStart Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDateRangeStart:



Public Member Functions

- [storeDateRangeStart](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.109.1 Detailed Description

Store the start of the date range.

Definition at line 53 of file [ScheduleParserHelper.hpp](#).

24.109.2 Constructor & Destructor Documentation

24.109.2.1 AIRINV::ScheduleParserHelper::storeDateRangeStart::storeDateRangeStart (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 61 of file [ScheduleParserHelper.cpp](#).

24.109.3 Member Function Documentation

24.109.3.1 void AIRINV::ScheduleParserHelper::storeDateRangeStart::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 66 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_dateRangeStart](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itSeconds](#), and [AIRINV::FlightPeriodStruct::getDate\(\)](#).

24.109.4 Member Data Documentation

24.109.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

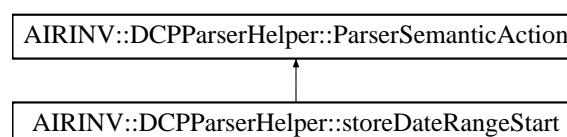
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.110 AIRINV::DCPParserHelper::storeDateRangeStart Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDateRangeStart:



Public Member Functions

- [storeDateRangeStart](#) (DCPRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.110.1 Detailed Description

Store the parsed start of the date range.

Definition at line 68 of file [DCPParserHelper.hpp](#).

24.110.2 Constructor & Destructor Documentation

24.110.2.1 AIRINV::DCPParserHelper::storeDateRangeStart::storeDateRangeStart (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 86 of file [DCPParserHelper.cpp](#).

24.110.3 Member Function Documentation

24.110.3.1 void AIRINV::DCPParserHelper::storeDateRangeStart::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 91 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.110.4 Member Data Documentation

24.110.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

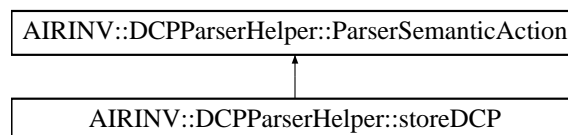
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.111 AIRINV::DCPParserHelper::storeDCP Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDCP:



Public Member Functions

- [storeDCP](#) (DCPRuleStruct &)
- void [operator\(\)](#) (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.111.1 Detailed Description

Store the parsed DCP value.

Definition at line 188 of file [DCPParserHelper.hpp](#).

24.111.2 Constructor & Destructor Documentation

24.111.2.1 AIRINV::DCPParserHelper::storeDCP::storeDCP (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 314 of file [DCPParserHelper.cpp](#).

24.111.3 Member Function Documentation

24.111.3.1 void AIRINV::DCPParserHelper::storeDCP::operator() (double iDCP, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 319 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.111.4 Member Data Documentation

24.111.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParser-](#)

Helper::storeStartRangeTime::operator(), AIRINV::DCPParserHelper::storeEndRangeTime::operator(), AIRINV::DCPParserHelper::storePOS::operator(), AIRINV::DCPParserHelper::storeCabinCode::operator(), AIRINV::DCPParserHelper::storeChannel::operator(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator(), AIRINV::DCPParserHelper::storeSaturdayStay::operator(), AIRINV::DCPParserHelper::storeChangeFees::operator(), AIRINV::DCPParserHelper::storeNonRefundable::operator(), AIRINV::DCPParserHelper::storeMinimumStay::operator(), operator(), AIRINV::DCPParserHelper::storeAirlineCode::operator(), AIRINV::DCPParserHelper::storeClass::operator(), and AIRINV::DCPParserHelper::doEndDCP::operator().

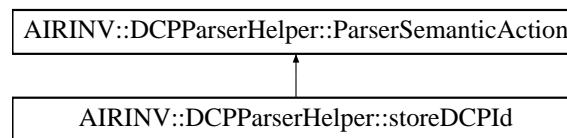
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.112 AIRINV::DCPParserHelper::storeDCPIId Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDCPIId:



Public Member Functions

- [storeDCPIId](#) (DCPRuleStruct &)
- void [operator\(\)](#) (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.112.1 Detailed Description

Store the parsed DCP Id.

Definition at line 38 of file [DCPParserHelper.hpp](#).

24.112.2 Constructor & Destructor Documentation

24.112.2.1 AIRINV::DCPParserHelper::storeDCPIId::storeDCPIId (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 30 of file [DCPParserHelper.cpp](#).

24.112.3 Member Function Documentation

24.112.3.1 void AIRINV::DCPParserHelper::storeDCPIId::operator() (unsigned int *iDCPIId*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 35 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.112.4 Member Data Documentation

24.112.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

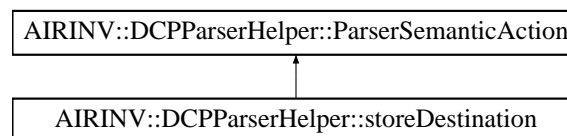
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.113 AIRINV::DCPParserHelper::storeDestination Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDestination:



Public Member Functions

- [storeDestination](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.113.1 Detailed Description

Store the parsed destination.

Definition at line 58 of file [DCPParserHelper.hpp](#).

24.113.2 Constructor & Destructor Documentation

24.113.2.1 AIRINV::DCPParserHelper::storeDestination::storeDestination (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 70 of file [DCPParserHelper.cpp](#).

24.113.3 Member Function Documentation

24.113.3.1 void AIRINV::DCPParserHelper::storeDestination::operator() (std::vector< char > iChar, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 75 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.113.4 Member Data Documentation

24.113.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPID::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

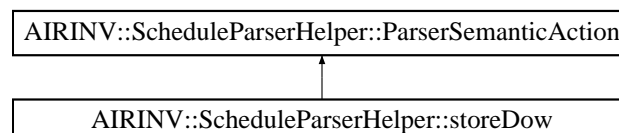
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.114 AIRINV::ScheduleParserHelper::storeDow Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDow:



Public Member Functions

- [storeDow](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.114.1 Detailed Description

Store the DOW (day of the Week).

Definition at line 69 of file [ScheduleParserHelper.hpp](#).

24.114.2 Constructor & Destructor Documentation

24.114.2.1 AIRINV::ScheduleParserHelper::storeDow::storeDow ([FlightPeriodStruct](#) & [ioFlightPeriod](#))

Actor Constructor.

Definition at line 99 of file [ScheduleParserHelper.cpp](#).

24.114.3 Member Function Documentation

24.114.3.1 void AIRINV::ScheduleParserHelper::storeDow::operator() ([iterator_t iStr](#), [iterator_t iStrEnd](#)) const

Actor Function (functor).

Definition at line 104 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_dow](#), and [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#).

24.114.4 Member Data Documentation

24.114.4.1 [FlightPeriodStruct](#)& [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#)
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

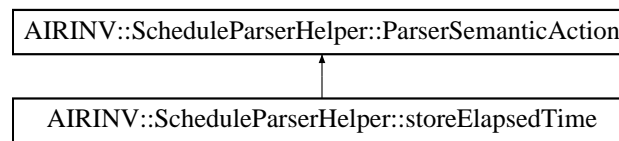
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.115 AIRINV::ScheduleParserHelper::storeElapsedTime Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeElapsedTime:



Public Member Functions

- [storeElapsedTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.115.1 Detailed Description

Store the elapsed time.

Definition at line 109 of file [ScheduleParserHelper.hpp](#).

24.115.2 Constructor & Destructor Documentation

24.115.2.1 AIRINV::ScheduleParserHelper::storeElapsedTime::storeElapsedTime ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 195 of file [ScheduleParserHelper.cpp](#).

24.115.3 Member Function Documentation

24.115.3.1 void AIRINV::ScheduleParserHelper::storeElapsedTime::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 200 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_dateOffset](#), [AIRINV::LegStruct::_elapsed](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_itSeconds](#), [AIRINV::LegStruct::_offDateOffset](#), and [AIRINV::FlightPeriodStruct::getTime\(\)](#).

24.115.4 Member Data Documentation

24.115.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#),

AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator(), AIRINV::ScheduleParserHelper::storeOffTime::operator(), operator(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator(), AIRINV::ScheduleParserHelper::storeCapacity::operator(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRINV::ScheduleParserHelper::storeClasses::operator(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator(), AIRINV::ScheduleParserHelper::storeFClasses::operator(), and AIRINV::ScheduleParserHelper::doEndFlight::operator().

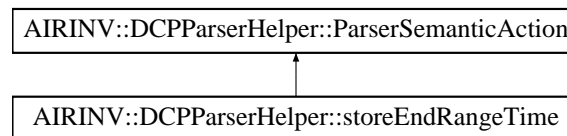
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.116 AIRINV::DCPParserHelper::storeEndRangeTime Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeEndRangeTime:



Public Member Functions

- [storeEndRangeTime](#) (DCPRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.116.1 Detailed Description

Store the parsed end start range time.

Definition at line 98 of file [DCPParserHelper.hpp](#).

24.116.2 Constructor & Destructor Documentation

24.116.2.1 AIRINV::DCPParserHelper::storeEndRangeTime::storeEndRangeTime (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 133 of file [DCPParserHelper.cpp](#).

24.116.3 Member Function Documentation

24.116.3.1 `void AIRINV::DCPParserHelper::storeEndRangeTime::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const`

Actor Function (functor).

Definition at line 138 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.116.4 Member Data Documentation

24.116.4.1 `DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule` [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPid::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

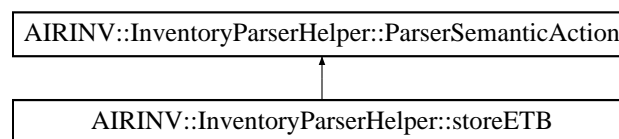
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.117 AIRINV::InventoryParserHelper::storeETB Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeETB:



Public Member Functions

- [storeETB](#) ([FlightDateStruct](#) &)
- `void operator() (double iReal) const`

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.117.1 Detailed Description

Store the parsed Expected To Board (ETB) number.

Definition at line 197 of file [InventoryParserHelper.hpp](#).

24.117.2 Constructor & Destructor Documentation

24.117.2.1 AIRINV::InventoryParserHelper::storeETB::storeETB (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 329 of file [InventoryParserHelper.cpp](#).

24.117.3 Member Function Documentation

24.117.3.1 void AIRINV::InventoryParserHelper::storeETB::operator() (double iReal) const

Actor Function (functor).

Definition at line 334 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_etb](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_itLegCabin](#).

24.117.4 Member Data Documentation

24.117.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailality::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

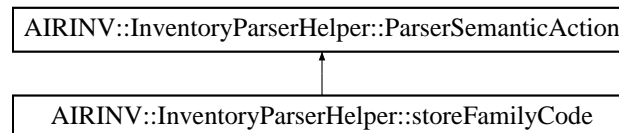
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.118 AIRINV::InventoryParserHelper::storeFamilyCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFamilyCode:



Public Member Functions

- [storeFamilyCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (int iCode) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.118.1 Detailed Description

Store the parsed family code.

Definition at line 409 of file [InventoryParserHelper.hpp](#).

24.118.2 Constructor & Destructor Documentation

24.118.2.1 AIRINV::InventoryParserHelper::storeFamilyCode::storeFamilyCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 705 of file [InventoryParserHelper.cpp](#).

24.118.3 Member Function Documentation

24.118.3.1 void AIRINV::InventoryParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

Definition at line 710 of file [InventoryParserHelper.cpp](#).

References [AIRINV::FareFamilyStruct::familyCode](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), and [AIRINV::FlightDateStruct::_itSegmentCabin](#).

24.118.4 Member Data Documentation

24.118.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

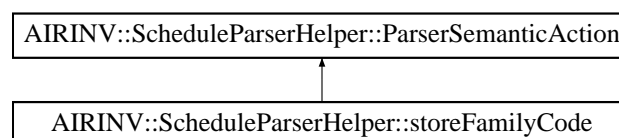
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.119 AIRINV::ScheduleParserHelper::storeFamilyCode Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFamilyCode:



Public Member Functions

- [storeFamilyCode](#) ([FlightPeriodStruct](#) &)

- void [operator\(\)](#) (int iCode) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.119.1 Detailed Description

Store the parsed family code.

Definition at line 176 of file [ScheduleParserHelper.hpp](#).

24.119.2 Constructor & Destructor Documentation

24.119.2.1 AIRINV::ScheduleParserHelper::storeFamilyCode::storeFamilyCode ([FlightPeriodStruct](#) & [ioFlightPeriod](#))

Actor Constructor.

Definition at line 335 of file [ScheduleParserHelper.cpp](#).

24.119.3 Member Function Documentation

24.119.3.1 void AIRINV::ScheduleParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

Definition at line 340 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FareFamilyStruct::_familyCode](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), and [AIRINV::FlightPeriodStruct::_itSegmentCabin](#).

24.119.4 Member Data Documentation

24.119.4.1 [FlightPeriodStruct](#)& [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#) [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

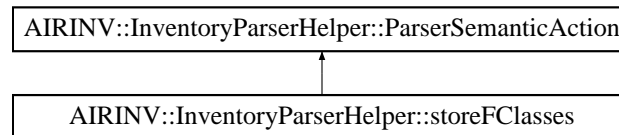
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.120 AIRINV::InventoryParserHelper::storeFClasses Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFClasses:



Public Member Functions

- [storeFClasses](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.120.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 417 of file [InventoryParserHelper.hpp](#).

24.120.2 Constructor & Destructor Documentation

24.120.2.1 AIRINV::InventoryParserHelper::storeFClasses::storeFClasses ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 717 of file [InventoryParserHelper.cpp](#).

24.120.3 Member Function Documentation

24.120.3.1 void AIRINV::InventoryParserHelper::storeFClasses::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 722 of file [InventoryParserHelper.cpp](#).

References [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::BookingClassStruct::_classCode](#), [AIRINV::FareFamilyStruct::_classes](#), [AIRINV::FareFamilyStruct::_classList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightDateStruct::_itSegment](#), and [AIRINV::FlightDateStruct::_itSegmentCabin](#).

24.120.4 Member Data Documentation

24.120.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

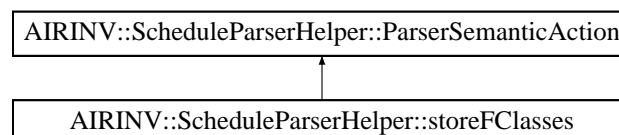
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.121 AIRINV::ScheduleParserHelper::storeFClasses Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFClasses:



Public Member Functions

- [storeFClasses](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.121.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 184 of file [ScheduleParserHelper.hpp](#).

24.121.2 Constructor & Destructor Documentation

24.121.2.1 AIRINV::ScheduleParserHelper::storeFClasses::storeFClasses (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 348 of file [ScheduleParserHelper.cpp](#).

24.121.3 Member Function Documentation

24.121.3.1 void AIRINV::ScheduleParserHelper::storeFClasses::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 353 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRINV::FareFamilyStruct::_familyCode](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightPeriodStruct::_itSegment](#), [AIRINV::FlightPeriodStruct::_itSegmentCabin](#), and [AIRINV::FlightPeriodStruct::addFareFamily\(\)](#).

24.121.4 Member Data Documentation

24.121.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

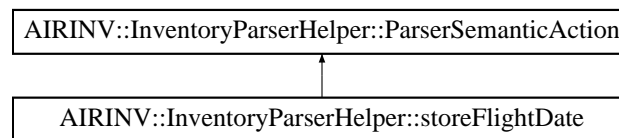
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.122 AIRINV::InventoryParserHelper::storeFlightDate Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightDate:



Public Member Functions

- [storeFlightDate](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.122.1 Detailed Description

Store the flight date.

Definition at line 61 of file [InventoryParserHelper.hpp](#).

24.122.2 Constructor & Destructor Documentation

24.122.2.1 AIRINV::InventoryParserHelper::storeFlightDate::storeFlightDate ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 80 of file [InventoryParserHelper.cpp](#).

24.122.3 Member Function Documentation

24.122.3.1 void AIRINV::InventoryParserHelper::storeFlightDate::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 85 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_flightDate](#), and [AIRINV::FlightDateStruct::getDate\(\)](#).

24.122.4 Member Data Documentation

24.122.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#).

[::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

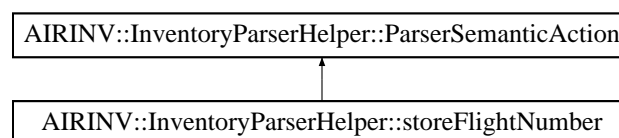
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.123 AIRINV::InventoryParserHelper::storeFlightNumber Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightNumber:



Public Member Functions

- [storeFlightNumber](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (unsigned int iNumber) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.123.1 Detailed Description

Store the parsed flight number.

Definition at line 53 of file [InventoryParserHelper.hpp](#).

24.123.2 Constructor & Destructor Documentation

24.123.2.1 AIRINV::InventoryParserHelper::storeFlightNumber::storeFlightNumber (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 70 of file [InventoryParserHelper.cpp](#).

24.123.3 Member Function Documentation

24.123.3.1 void AIRINV::InventoryParserHelper::storeFlightNumber::operator() (unsigned int iNumber) const

Actor Function (functor).

Definition at line 75 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), and [AIRINV::FlightDateStruct::_flightNumber](#).

24.123.4 Member Data Documentation

24.123.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

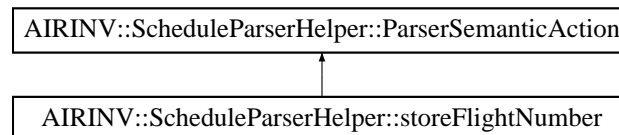
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.124 AIRINV::ScheduleParserHelper::storeFlightNumber Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFlightNumber:



Public Member Functions

- [storeFlightNumber](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (unsigned int iNumber) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.124.1 Detailed Description

Store the parsed flight number.

Definition at line 45 of file [ScheduleParserHelper.hpp](#).

24.124.2 Constructor & Destructor Documentation

24.124.2.1 AIRINV::ScheduleParserHelper::storeFlightNumber::storeFlightNumber ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 50 of file [ScheduleParserHelper.cpp](#).

24.124.3 Member Function Documentation

24.124.3.1 void AIRINV::ScheduleParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 55 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_flightNumber](#), and [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#).

24.124.4 Member Data Documentation

24.124.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

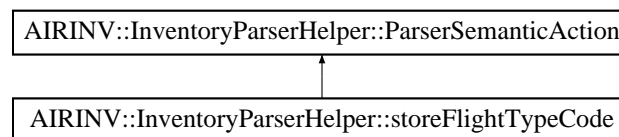
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.125 AIRINV::InventoryParserHelper::storeFlightTypeCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightTypeCode:



Public Member Functions

- [storeFlightTypeCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.125.1 Detailed Description

Store the flight type code.

Definition at line 69 of file [InventoryParserHelper.hpp](#).

24.125.2 Constructor & Destructor Documentation

24.125.2.1 AIRINV::InventoryParserHelper::storeFlightTypeCode::storeFlightTypeCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 91 of file [InventoryParserHelper.cpp](#).

24.125.3 Member Function Documentation

24.125.3.1 void AIRINV::InventoryParserHelper::storeFlightTypeCode::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 96 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_flightTypeCode](#), and [AIRINV::FlightTypeCode::getCode\(\)](#).

24.125.4 Member Data Documentation

24.125.4.1 **FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate**
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

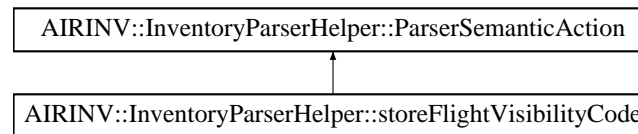
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.126 AIRINV::InventoryParserHelper::storeFlightVisibilityCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightVisibilityCode:



Public Member Functions

- [storeFlightVisibilityCode](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.126.1 Detailed Description

Store the flight visibility code.

Definition at line 77 of file [InventoryParserHelper.hpp](#).

24.126.2 Constructor & Destructor Documentation

24.126.2.1 AIRINV::InventoryParserHelper::storeFlightVisibilityCode::storeFlightVisibilityCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 106 of file [InventoryParserHelper.cpp](#).

24.126.3 Member Function Documentation

24.126.3.1 void AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 111 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_flightVisibilityCode](#), and [AIRINV::FlightVisibilityCode::getCode\(\)](#).

24.126.4 Member Data Documentation

24.126.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::](#)

[::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

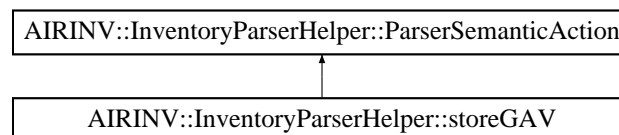
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.127 AIRINV::InventoryParserHelper::storeGAV Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeGAV:



Public Member Functions

- [storeGAV](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.127.1 Detailed Description

Store the parsed Gross Availability (GAV).

Definition at line 181 of file [InventoryParserHelper.hpp](#).

24.127.2 Constructor & Destructor Documentation

24.127.2.1 AIRINV::InventoryParserHelper::storeGAV::storeGAV (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 307 of file [InventoryParserHelper.cpp](#).

24.127.3 Member Function Documentation

24.127.3.1 void AIRINV::InventoryParserHelper::storeGAV::operator() (double iReal) const

Actor Function (functor).

Definition at line 312 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::LegCabinStruct::_gav](#), and [AIRINV::FlightDateStruct::_itLegCabin](#).

24.127.4 Member Data Documentation

24.127.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

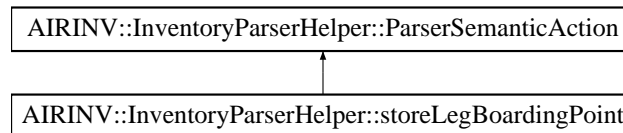
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.128 AIRINV::InventoryParserHelper::storeLegBoardingPoint Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegBoardingPoint:



Public Member Functions

- [storeLegBoardingPoint](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.128.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 85 of file [InventoryParserHelper.hpp](#).

24.128.2 Constructor & Destructor Documentation

24.128.2.1 AIRINV::InventoryParserHelper::storeLegBoardingPoint::storeLegBoardingPoint ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 121 of file [InventoryParserHelper.cpp](#).

24.128.3 Member Function Documentation

24.128.3.1 void AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 126 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingPoint](#), [AIRINV::LegCabinStruct::_bucketList](#), [AIRINV::LegCabinStruct::_cabinCode](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBucket](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itLegCabin](#), [AIRINV::FlightDateStruct::_legList](#), [AIRINV::BucketStruct::_yieldRangeUpperValue](#), and [AIRINV::FlightDateStruct::addAirport\(\)](#).

24.128.4 Member Data Documentation

24.128.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

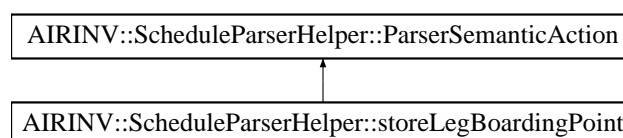
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.129 AIRINV::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegBoardingPoint:



Public Member Functions

- [storeLegBoardingPoint](#) ([FlightPeriodStruct](#) &)

- void [operator\(\)](#) (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.129.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 77 of file [ScheduleParserHelper.hpp](#).

24.129.2 Constructor & Destructor Documentation

24.129.2.1 AIRINV::ScheduleParserHelper::storeLegBoardingPoint::storeLegBoardingPoint (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 111 of file [ScheduleParserHelper.cpp](#).

24.129.3 Member Function Documentation

24.129.3.1 void AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 116 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingPoint](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_legAlreadyDefined](#), [AIRINV::FlightPeriodStruct::_legList](#), and [AIRINV::FlightPeriodStruct::addAirport\(\)](#).

24.129.4 Member Data Documentation

24.129.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

The documentation for this struct was generated from the following files:

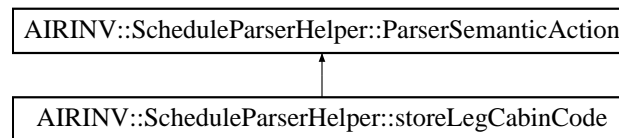
- [airinv/command/ScheduleParserHelper.hpp](#)

- [airinv/command/ScheduleParserHelper.cpp](#)

24.130 AIRINV::ScheduleParserHelper::storeLegCabinCode Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegCabinCode:



Public Member Functions

- [storeLegCabinCode](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.130.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 117 of file [ScheduleParserHelper.hpp](#).

24.130.2 Constructor & Destructor Documentation

24.130.2.1 AIRINV::ScheduleParserHelper::storeLegCabinCode::storeLegCabinCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 216 of file [ScheduleParserHelper.cpp](#).

24.130.3 Member Function Documentation

24.130.3.1 void AIRINV::ScheduleParserHelper::storeLegCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 221 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_cabinCode](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRINV::FlightPeriodStruct::_itLegCabin](#).

24.130.4 Member Data Documentation

24.130.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

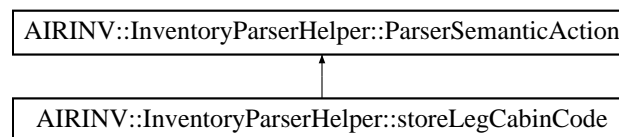
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.131 AIRINV::InventoryParserHelper::storeLegCabinCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegCabinCode:



Public Member Functions

- [storeLegCabinCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.131.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 133 of file [InventoryParserHelper.hpp](#).

24.131.2 Constructor & Destructor Documentation

24.131.2.1 AIRINV::InventoryParserHelper::storeLegCabinCode::storeLegCabinCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 223 of file [InventoryParserHelper.cpp](#).

24.131.3 Member Function Documentation

24.131.3.1 void AIRINV::InventoryParserHelper::storeLegCabinCode::operator() (char iChar) const

Actor Function (functor).

Definition at line 228 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_bucketList](#), [AIRINV::LegCabinStruct::_cabinCode](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBucket](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::BucketStruct::_yieldRangeUpperValue](#).

24.131.4 Member Data Documentation

24.131.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailablity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

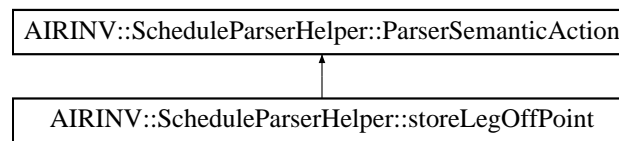
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.132 AIRINV::ScheduleParserHelper::storeLegOffPoint Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegOffPoint:



Public Member Functions

- [storeLegOffPoint](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.132.1 Detailed Description

Store the parsed leg off point.

Definition at line 85 of file [ScheduleParserHelper.hpp](#).

24.132.2 Constructor & Destructor Documentation

24.132.2.1 AIRINV::ScheduleParserHelper::storeLegOffPoint::storeLegOffPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 140 of file [ScheduleParserHelper.cpp](#).

24.132.3 Member Function Documentation

24.132.3.1 void AIRINV::ScheduleParserHelper::storeLegOffPoint::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 145 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::LegStruct::_offPoint](#), and [AIRINV::FlightPeriodStruct::addAirport\(\)](#).

24.132.4 Member Data Documentation

24.132.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#),

[AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

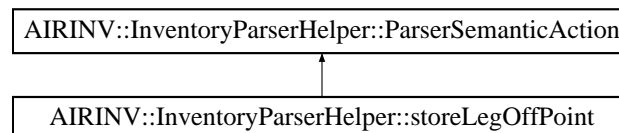
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.133 AIRINV::InventoryParserHelper::storeLegOffPoint Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegOffPoint:



Public Member Functions

- [storeLegOffPoint](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.133.1 Detailed Description

Store the parsed leg off point.

Definition at line 93 of file [InventoryParserHelper.hpp](#).

24.133.2 Constructor & Destructor Documentation

24.133.2.1 AIRINV::InventoryParserHelper::storeLegOffPoint::storeLegOffPoint ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 157 of file [InventoryParserHelper.cpp](#).

24.133.3 Member Function Documentation

24.133.3.1 void AIRINV::InventoryParserHelper::storeLegOffPoint::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 162 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::LegStruct::_offPoint](#), and [AIRINV::FlightDateStruct::addAirport\(\)](#).

24.133.4 Member Data Documentation

24.133.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

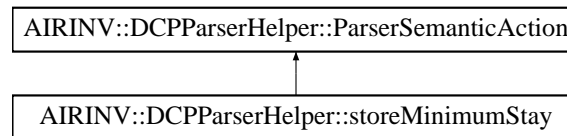
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.134 AIRINV::DCPParserHelper::storeMinimumStay Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeMinimumStay:



Public Member Functions

- [storeMinimumStay](#) (DCPRuleStruct &)
- void [operator\(\)](#) (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.134.1 Detailed Description

Store the parsed minimum stay.

Definition at line 178 of file [DCPParserHelper.hpp](#).

24.134.2 Constructor & Destructor Documentation

24.134.2.1 AIRINV::DCPParserHelper::storeMinimumStay::storeMinimumStay (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 299 of file [DCPParserHelper.cpp](#).

24.134.3 Member Function Documentation

24.134.3.1 void AIRINV::DCPParserHelper::storeMinimumStay::operator() (unsigned int iMinStay, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 304 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.134.4 Member Data Documentation

24.134.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#),

[AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

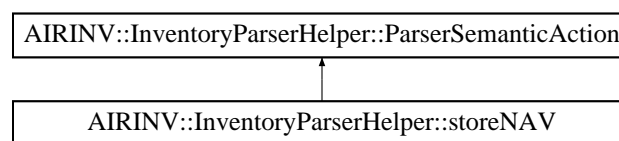
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.135 AIRINV::InventoryParserHelper::storeNAV Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNAV:



Public Member Functions

- [storeNAV](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.135.1 Detailed Description

Store the parsed Net Availability (NAV).

Definition at line 173 of file [InventoryParserHelper.hpp](#).

24.135.2 Constructor & Destructor Documentation

24.135.2.1 AIRINV::InventoryParserHelper::storeNAV::storeNAV ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 296 of file [InventoryParserHelper.cpp](#).

24.135.3 Member Function Documentation

24.135.3.1 void AIRINV::InventoryParserHelper::storeNAV::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 301 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::LegCabinStruct::_nav](#).

24.135.4 Member Data Documentation

24.135.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

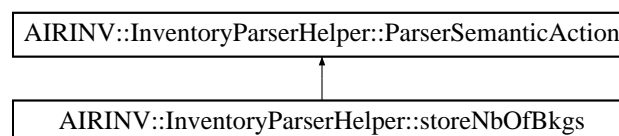
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.136 AIRINV::InventoryParserHelper::storeNbOfBkgs Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfBkgs:



Public Member Functions

- [storeNbOfBkgs](#) ([FlightDateStruct](#) &)

- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.136.1 Detailed Description

Store the parsed number of bookings (at booking class level).

Definition at line 333 of file [InventoryParserHelper.hpp](#).

24.136.2 Constructor & Destructor Documentation

24.136.2.1 AIRINV::InventoryParserHelper::storeNbOfBkgs::storeNbOfBkgs ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 602 of file [InventoryParserHelper.cpp](#).

24.136.3 Member Function Documentation

24.136.3.1 void AIRINV::InventoryParserHelper::storeNbOfBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 607 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_nbOfBookings](#).

24.136.4 Member Data Documentation

24.136.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode](#)

[::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

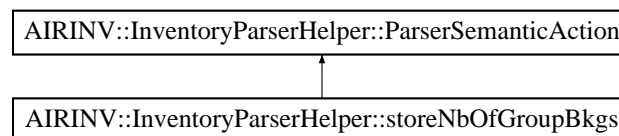
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.137 AIRINV::InventoryParserHelper::storeNbOfGroupBkgs Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfGroupBkgs:



Public Member Functions

- [storeNbOfGroupBkgs](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.137.1 Detailed Description

Store the parsed number of group bookings (at booking class level).

Definition at line 341 of file [InventoryParserHelper.hpp](#).

24.137.2 Constructor & Destructor Documentation

24.137.2.1 AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::storeNbOfGroupBkgs ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 613 of file [InventoryParserHelper.cpp](#).

24.137.3 Member Function Documentation

24.137.3.1 void AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 618 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_nbOfGroupBookings](#).

24.137.4 Member Data Documentation

24.137.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

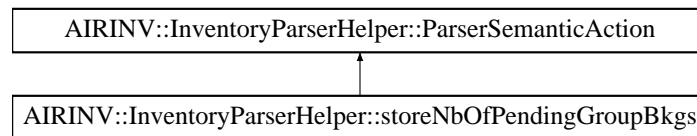
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.138 AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs:



Public Member Functions

- [storeNbOfPendingGroupBkgs](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.138.1 Detailed Description

Store the parsed number of pending group bookings (at booking class level).

Definition at line 349 of file [InventoryParserHelper.hpp](#).

24.138.2 Constructor & Destructor Documentation

24.138.2.1 AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::storeNbOfPendingGroupBkgs ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 625 of file [InventoryParserHelper.cpp](#).

24.138.3 Member Function Documentation

24.138.3.1 void AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 630 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_it-BookingClass](#), and [AIRINV::BookingClassStruct::_nbOfPendingGroupBookings](#).

24.138.4 Member Data Documentation

24.138.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::](#)

[::storeOffTime::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#)(), [operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#)(), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#)(), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#)).

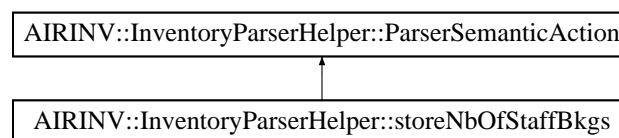
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.139 AIRINV::InventoryParserHelper::storeNbOfStaffBkgs Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfStaffBkgs:



Public Member Functions

- [storeNbOfStaffBkgs](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.139.1 Detailed Description

Store the parsed number of staff bookings (at booking class level).

Definition at line 357 of file [InventoryParserHelper.hpp](#).

24.139.2 Constructor & Destructor Documentation

24.139.2.1 AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::storeNbOfStaffBkgs (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 636 of file [InventoryParserHelper.cpp](#).

24.139.3 Member Function Documentation

24.139.3.1 void AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator() (double iReal) const

Actor Function (functor).

Definition at line 641 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_nbOfStaffBookings](#).

24.139.4 Member Data Documentation

24.139.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

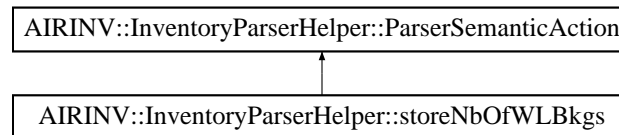
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.140 AIRINV::InventoryParserHelper::storeNbOfWLBkgs Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfWLBkgs:



Public Member Functions

- [storeNbOfWLBkgs](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.140.1 Detailed Description

Store the parsed number of wait-list bookings (at booking class level).

Definition at line 366 of file [InventoryParserHelper.hpp](#).

24.140.2 Constructor & Destructor Documentation

24.140.2.1 AIRINV::InventoryParserHelper::storeNbOfWLBkgs::storeNbOfWLBkgs ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 647 of file [InventoryParserHelper.cpp](#).

24.140.3 Member Function Documentation

24.140.3.1 void AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 652 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_it-BookingClass](#), and [AIRINV::BookingClassStruct::_nbOfWLBBookings](#).

24.140.4 Member Data Documentation

24.140.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGA::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

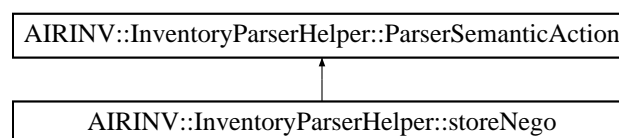
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.141 AIRINV::InventoryParserHelper::storeNego Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNego:



Public Member Functions

- [storeNego](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.141.1 Detailed Description

Store the negotiated allotment (at booking class level).

Definition at line 309 of file [InventoryParserHelper.hpp](#).

24.141.2 Constructor & Destructor Documentation

24.141.2.1 AIRINV::InventoryParserHelper::storeNego::storeNego (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 569 of file [InventoryParserHelper.cpp](#).

24.141.3 Member Function Documentation

24.141.3.1 void AIRINV::InventoryParserHelper::storeNego::operator() (double iReal) const

Actor Function (functor).

Definition at line 574 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_nego](#).

24.141.4 Member Data Documentation

24.141.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB-](#)

[::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

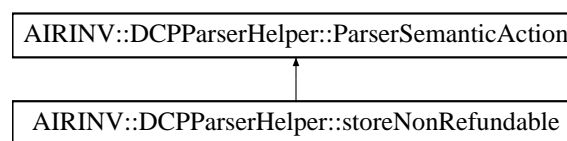
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.142 AIRINV::DCPParserHelper::storeNonRefundable Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeNonRefundable:



Public Member Functions

- [storeNonRefundable](#) (DCPRuleStruct &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.142.1 Detailed Description

Store the parsed refundable option

Definition at line 168 of file [DCPParserHelper.hpp](#).

24.142.2 Constructor & Destructor Documentation

24.142.2.1 **AIRINV::DCPParserHelper::storeNonRefundable::storeNonRefundable** (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 274 of file [DCPParserHelper.cpp](#).

24.142.3 Member Function Documentation

24.142.3.1 void AIRINV::DCPParserHelper::storeNonRefundable::operator() (char *iNonRefundable*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 279 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.142.4 Member Data Documentation

24.142.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPid::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

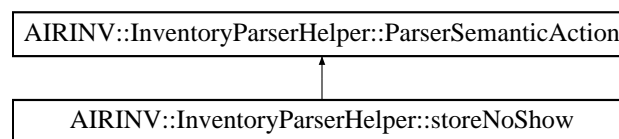
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.143 AIRINV::InventoryParserHelper::storeNoShow Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeNoShow:



Public Member Functions

- [storeNoShow](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.143.1 Detailed Description

Store the parsed No-Show percentage (at booking class level).

Definition at line 317 of file [InventoryParserHelper.hpp](#).

24.143.2 Constructor & Destructor Documentation

24.143.2.1 AIRINV::InventoryParserHelper::storeNoShow::storeNoShow ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 580 of file [InventoryParserHelper.cpp](#).

24.143.3 Member Function Documentation

24.143.3.1 void AIRINV::InventoryParserHelper::storeNoShow::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 585 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_it-BookingClass, and AIRINV::BookingClassStruct::_noShowPercentage.

24.143.4 Member Data Documentation

24.143.4.1 **FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate**
[inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator(), AIRINV::InventoryParserHelper::storeAirlineCode::operator(), AIRINV::InventoryParserHelper::storeFlightNumber::operator(), AIRINV::InventoryParserHelper::storeFlightDate::operator(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator(), AIRINV::InventoryParserHelper::storeBoardingDate::operator(), AIRINV::InventoryParserHelper::storeBoardingTime::operator(), AIRINV::InventoryParserHelper::storeOffDate::operator(), AIRINV::InventoryParserHelper::storeOffTime::operator(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator(), AIRINV::InventoryParserHelper::storeAU::operator(), AIRINV::InventoryParserHelper::storeUPR::operator(), AIRINV::InventoryParserHelper::storeBookingCounter::operator(), AIRINV::InventoryParserHelper::storeNAV::operator(), AIRINV::InventoryParserHelper::storeGAV::operator(), AIRINV::InventoryParserHelper::storeACP::operator(), AIRINV::InventoryParserHelper::storeETB::operator(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator(), AIRINV::InventoryParserHelper::storeBucketAvailablity::operator(), AIRINV::InventoryParserHelper::storeSeatIndex::operator(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator(), AIRINV::InventoryParserHelper::storeClassCode::operator(), AIRINV::InventoryParserHelper::storeSubclassCode::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFCClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

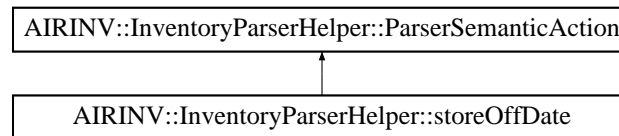
The documentation for this struct was generated from the following files:

- `airinv/command/InventoryParserHelper.hpp`
- `airinv/command/InventoryParserHelper.cpp`

24.144 AIRINV::InventoryParserHelper::storeOffDate Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeOffDate:



Public Member Functions

- [storeOffDate](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.144.1 Detailed Description

Store the off date.

Definition at line 117 of file [InventoryParserHelper.hpp](#).

24.144.2 Constructor & Destructor Documentation

24.144.2.1 AIRINV::InventoryParserHelper::storeOffDate::storeOffDate ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 200 of file [InventoryParserHelper.cpp](#).

24.144.3 Member Function Documentation

24.144.3.1 void AIRINV::InventoryParserHelper::storeOffDate::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 205 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::LegStruct::_offDate](#), and [AIRINV::FlightDateStruct::getDate\(\)](#).

24.144.4 Member Data Documentation

24.144.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#).

AIRINV::InventoryParserHelper::storeAU::operator(), AIRINV::InventoryParserHelper::storeUPR::operator(), AIRINV::InventoryParserHelper::storeBookingCounter::operator(), AIRINV::InventoryParserHelper::storeNAV::operator(), AIRINV::InventoryParserHelper::storeGAV::operator(), AIRINV::InventoryParserHelper::storeACP::operator(), AIRINV::InventoryParserHelper::storeETB::operator(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator(), AIRINV::InventoryParserHelper::storeBucketAvailability::operator(), AIRINV::InventoryParserHelper::storeSeatIndex::operator(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator(), AIRINV::InventoryParserHelper::storeClassCode::operator(), AIRINV::InventoryParserHelper::storeSubclassCode::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

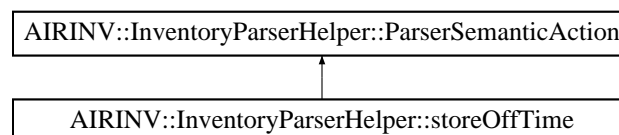
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.145 AIRINV::InventoryParserHelper::storeOffTime Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeOffTime:



Public Member Functions

- [storeOffTime](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.145.1 Detailed Description

Store the off time.

Definition at line 125 of file [InventoryParserHelper.hpp](#).

24.145.2 Constructor & Destructor Documentation

24.145.2.1 AIRINV::InventoryParserHelper::storeOffTime::storeOffTime (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 210 of file [InventoryParserHelper.cpp](#).

24.145.3 Member Function Documentation

24.145.3.1 void AIRINV::InventoryParserHelper::storeOffTime::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 215 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itSeconds](#), [AIRINV::LegStruct::_offTime](#), and [AIRINV::FlightDateStruct::getTime\(\)](#).

24.145.4 Member Data Documentation

24.145.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

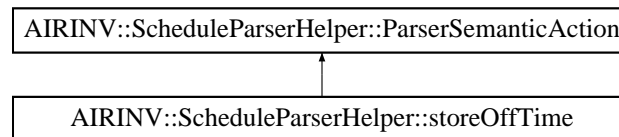
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.146 AIRINV::ScheduleParserHelper::storeOffTime Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeOffTime:



Public Member Functions

- [storeOffTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.146.1 Detailed Description

Store the off time.

Definition at line 101 of file [ScheduleParserHelper.hpp](#).

24.146.2 Constructor & Destructor Documentation

24.146.2.1 AIRINV::ScheduleParserHelper::storeOffTime::storeOffTime ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 174 of file [ScheduleParserHelper.cpp](#).

24.146.3 Member Function Documentation

24.146.3.1 void AIRINV::ScheduleParserHelper::storeOffTime::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 179 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::LegStruct::_boardingDateOffset](#), [AIRINV::FlightPeriodStruct::_dateOffset](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itLeg](#), [AIRINV::FlightPeriodStruct::_itSeconds](#), [AIRINV::LegStruct::_offTime](#), and [AIRINV::FlightPeriodStruct::getTime\(\)](#).

24.146.4 Member Data Documentation

24.146.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

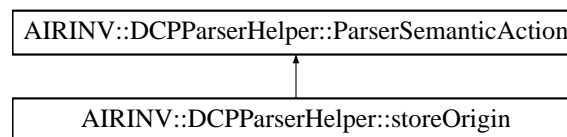
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.147 AIRINV::DCPParserHelper::storeOrigin Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeOrigin:



Public Member Functions

- [storeOrigin](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.147.1 Detailed Description

Store the parsed origin.

Definition at line 48 of file [DCPParserHelper.hpp](#).

24.147.2 Constructor & Destructor Documentation

24.147.2.1 AIRINV::DCPParserHelper::storeOrigin::storeOrigin (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 54 of file [DCPParserHelper.cpp](#).

24.147.3 Member Function Documentation

24.147.3.1 void AIRINV::DCPParserHelper::storeOrigin::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 59 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.147.4 Member Data Documentation

24.147.4.1 [DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#) [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPId::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)\(\)](#).

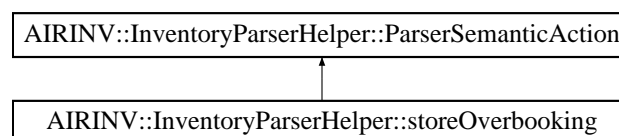
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.148 AIRINV::InventoryParserHelper::storeOverbooking Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeOverbooking:



Public Member Functions

- [storeOverbooking](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.148.1 Detailed Description

Store the parsed Overbooking percentage (at booking class level).

Definition at line 325 of file [InventoryParserHelper.hpp](#).

24.148.2 Constructor & Destructor Documentation

24.148.2.1 AIRINV::InventoryParserHelper::storeOverbooking::storeOverbooking (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 591 of file [InventoryParserHelper.cpp](#).

24.148.3 Member Function Documentation

24.148.3.1 void AIRINV::InventoryParserHelper::storeOverbooking::operator() (double iReal) const

Actor Function (functor).

Definition at line 596 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_overbookingPercentage](#).

24.148.4 Member Data Documentation

24.148.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::Inventory](#)

[ParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

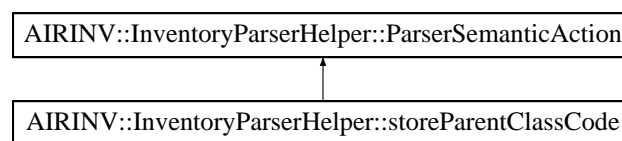
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.149 AIRINV::InventoryParserHelper::storeParentClassCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeParentClassCode:



Public Member Functions

- [storeParentClassCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.149.1 Detailed Description

Store the parsed class code of the parent sub-class.

Definition at line 277 of file [InventoryParserHelper.hpp](#).

24.149.2 Constructor & Destructor Documentation

24.149.2.1 AIRINV::InventoryParserHelper::storeParentClassCode::storeParentClassCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 523 of file [InventoryParserHelper.cpp](#).

24.149.3 Member Function Documentation

24.149.3.1 void AIRINV::InventoryParserHelper::storeParentClassCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 528 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_it-BookingClass](#), and [AIRINV::BookingClassStruct::_parentClassCode](#).

24.149.4 Member Data Documentation

24.149.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

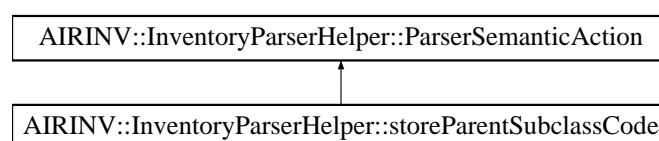
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.150 AIRINV::InventoryParserHelper::storeParentSubclassCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeParentSubclassCode:



Public Member Functions

- [storeParentSubclassCode](#) ([FlightDateStruct](#) &)

- void [operator\(\)](#) (unsigned int *iNumber*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.150.1 Detailed Description

Store the parsed sub-class code of the parent sub-class.

Definition at line 285 of file [InventoryParserHelper.hpp](#).

24.150.2 Constructor & Destructor Documentation

24.150.2.1 AIRINV::InventoryParserHelper::storeParentSubclassCode::storeParentSubclassCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 535 of file [InventoryParserHelper.cpp](#).

24.150.3 Member Function Documentation

24.150.3.1 void AIRINV::InventoryParserHelper::storeParentSubclassCode::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 540 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_parentSubclassCode](#).

24.150.4 Member Data Documentation

24.150.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode](#)

[::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

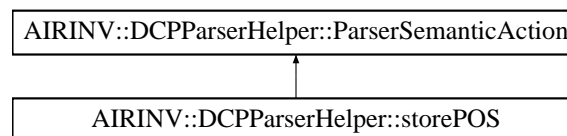
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.151 AIRINV::DCPParserHelper::storePOS Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storePOS:



Public Member Functions

- [storePOS](#) (DCPRuleStruct &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.151.1 Detailed Description

Store the parsed customer position.

Definition at line 108 of file [DCPParserHelper.hpp](#).

24.151.2 Constructor & Destructor Documentation

24.151.2.1 AIRINV::DCPParserHelper::storePOS::storePOS (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 150 of file [DCPParserHelper.cpp](#).

24.151.3 Member Function Documentation

24.151.3.1 void AIRINV::DCPParserHelper::storePOS::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 155 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.151.4 Member Data Documentation

24.151.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPID::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

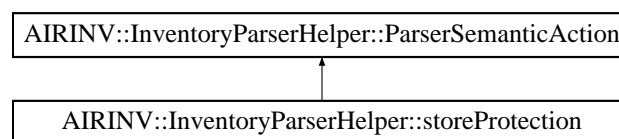
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.152 AIRINV::InventoryParserHelper::storeProtection Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeProtection:



Public Member Functions

- [storeProtection](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.152.1 Detailed Description

Store the parsed protection (at booking class level).

Definition at line 301 of file [InventoryParserHelper.hpp](#).

24.152.2 Constructor & Destructor Documentation

24.152.2.1 AIRINV::InventoryParserHelper::storeProtection::storeProtection (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 558 of file [InventoryParserHelper.cpp](#).

24.152.3 Member Function Documentation

24.152.3.1 void AIRINV::InventoryParserHelper::storeProtection::operator() (double iReal) const

Actor Function (functor).

Definition at line 563 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_protection](#).

24.152.4 Member Data Documentation

24.152.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailality::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#),

[AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

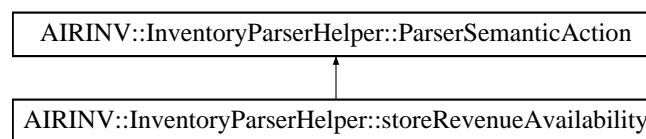
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.153 AIRINV::InventoryParserHelper::storeRevenueAvailability Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeRevenueAvailability:



Public Member Functions

- [storeRevenueAvailability](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.153.1 Detailed Description

Store the parsed number of net revenue availability (at booking class level).

Definition at line 401 of file [InventoryParserHelper.hpp](#).

24.153.2 Constructor & Destructor Documentation

24.153.2.1 AIRINV::InventoryParserHelper::storeRevenueAvailability::storeRevenueAvailability ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 694 of file [InventoryParserHelper.cpp](#).

24.153.3 Member Function Documentation

24.153.3.1 void AIRINV::InventoryParserHelper::storeRevenueAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 699 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_netRevenueAvailability](#).

24.153.4 Member Data Documentation

24.153.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

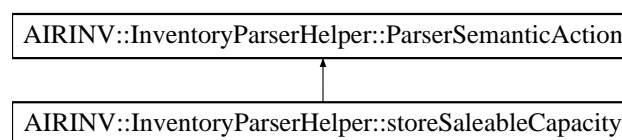
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.154 AIRINV::InventoryParserHelper::storeSaleableCapacity Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSaleableCapacity:



Public Member Functions

- [storeSaleableCapacity](#) ([FlightDateStruct](#) &)

- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.154.1 Detailed Description

Store the parsed saleable capacity.

Definition at line 141 of file [InventoryParserHelper.hpp](#).

24.154.2 Constructor & Destructor Documentation

24.154.2.1 AIRINV::InventoryParserHelper::storeSaleableCapacity::storeSaleableCapacity ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 252 of file [InventoryParserHelper.cpp](#).

24.154.3 Member Function Documentation

24.154.3.1 void AIRINV::InventoryParserHelper::storeSaleableCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 257 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::LegCabinStruct::_saleableCapacity](#).

24.154.4 Member Data Documentation

24.154.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailality::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclass](#)

Code::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFCClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

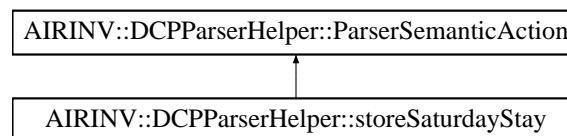
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.155 AIRINV::DCPParserHelper::storeSaturdayStay Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeSaturdayStay:



Public Member Functions

- [storeSaturdayStay](#) (DCPRuleStruct &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.155.1 Detailed Description

Store the parsed saturday night.

Definition at line 148 of file [DCPParserHelper.hpp](#).

24.155.2 Constructor & Destructor Documentation

24.155.2.1 AIRINV::DCPParserHelper::storeSaturdayStay::storeSaturdayStay (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 223 of file [DCPParserHelper.cpp](#).

24.155.3 Member Function Documentation

24.155.3.1 void AIRINV::DCPParserHelper::storeSaturdayStay::operator() (char *iSaturdayStay*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 228 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.155.4 Member Data Documentation

24.155.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPid::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::DCPParserHelper::storeStartRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

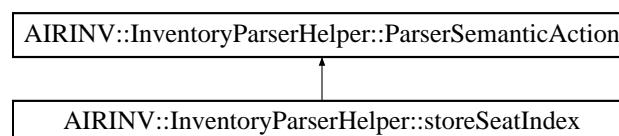
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.156 AIRINV::InventoryParserHelper::storeSeatIndex Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSeatIndex:



Public Member Functions

- [storeSeatIndex](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.156.1 Detailed Description

Store the parsed leg-cabin seat index.

Definition at line 221 of file [InventoryParserHelper.hpp](#).

24.156.2 Constructor & Destructor Documentation

24.156.2.1 AIRINV::InventoryParserHelper::storeSeatIndex::storeSeatIndex (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 371 of file [InventoryParserHelper.cpp](#).

24.156.3 Member Function Documentation

24.156.3.1 void AIRINV::InventoryParserHelper::storeSeatIndex::operator() (double iReal) const

Actor Function (functor).

Definition at line 376 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBucket](#), and [AIRINV::BucketStruct::_seatIndex](#).

24.156.4 Member Data Documentation

24.156.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB-](#)

[::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

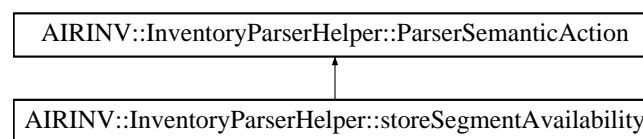
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.157 AIRINV::InventoryParserHelper::storeSegmentAvailability Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentAvailability:



Public Member Functions

- [storeSegmentAvailability](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.157.1 Detailed Description

Store the parsed number of segment availability (at booking class level).

Definition at line 392 of file [InventoryParserHelper.hpp](#).

24.157.2 Constructor & Destructor Documentation

24.157.2.1 AIRINV::InventoryParserHelper::storeSegmentAvailability::storeSegmentAvailability ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 682 of file [InventoryParserHelper.cpp](#).

24.157.3 Member Function Documentation

24.157.3.1 void AIRINV::InventoryParserHelper::storeSegmentAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 687 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_it-BookingClass](#), and [AIRINV::BookingClassStruct::_segmentAvailability](#).

24.157.4 Member Data Documentation

24.157.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailibility::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

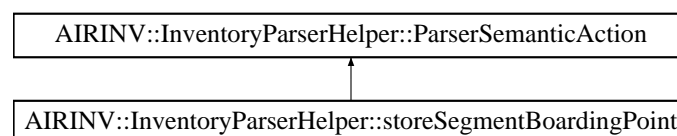
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.158 AIRINV::InventoryParserHelper::storeSegmentBoardingPoint Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentBoardingPoint:



Public Member Functions

- [storeSegmentBoardingPoint](#) ([FlightDateStruct](#) &)

- void [operator\(\)](#) (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.158.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 229 of file [InventoryParserHelper.hpp](#).

24.158.2 Constructor & Destructor Documentation

24.158.2.1 AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 383 of file [InventoryParserHelper.cpp](#).

24.158.3 Member Function Documentation

24.158.3.1 void AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 388 of file [InventoryParserHelper.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::LegCabinStruct::_bucketList](#), [AIRINV::LegCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::LegStruct::_cabinList](#), [AIRINV::BookingClassStruct::_classCode](#), [AIRINV::FareFamilyStruct::_classList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), [AIRINV::FlightDateStruct::_itBucket](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightDateStruct::_itLeg](#), [AIRINV::FlightDateStruct::_itLegCabin](#), [AIRINV::FlightDateStruct::_itSegment](#), [AIRINV::FlightDateStruct::_itSegmentCabin](#), [AIRINV::FlightDateStruct::_legList](#), and [AIRINV::FlightDateStruct::_segmentList](#).

24.158.4 Member Data Documentation

24.158.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGA-](#)

V::operator(), AIRINV::InventoryParserHelper::storeACP::operator(), AIRINV::InventoryParserHelper::storeETB::operator(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator(), AIRINV::InventoryParserHelper::storeBucketAvailability::operator(), AIRINV::InventoryParserHelper::storeSeatIndex::operator(), operator(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator(), AIRINV::InventoryParserHelper::storeClassCode::operator(), AIRINV::InventoryParserHelper::storeSubclassCode::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

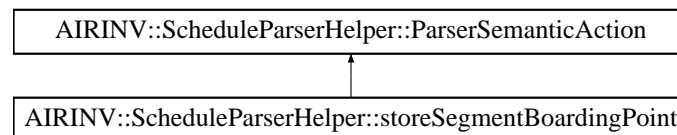
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.159 AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint:



Public Member Functions

- [storeSegmentBoardingPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.159.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 144 of file [ScheduleParserHelper.hpp](#).

24.159.2 Constructor & Destructor Documentation

24.159.2.1 AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 273 of file [ScheduleParserHelper.cpp](#).

24.159.3 Member Function Documentation

24.159.3.1 void AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 278 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::SegmentStruct::_boardingPoint](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRINV::FlightPeriodStruct::_itSegment](#).

24.159.4 Member Data Documentation

24.159.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

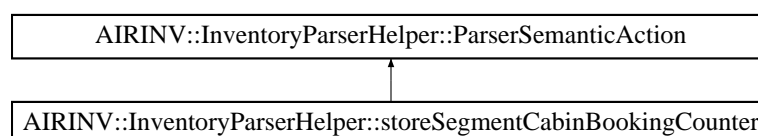
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.160 AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter:



Public Member Functions

- [storeSegmentCabinBookingCounter](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.160.1 Detailed Description

Store the parsed segment cabin number of bookings.

Definition at line 253 of file [InventoryParserHelper.hpp](#).

24.160.2 Constructor & Destructor Documentation

24.160.2.1 AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::storeSegmentCabinBookingCounter ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 481 of file [InventoryParserHelper.cpp](#).

24.160.3 Member Function Documentation

24.160.3.1 void AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 486 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itSegmentCabin](#), and [AIRINV::SegmentCabinStruct::_nbOfBookings](#).

24.160.4 Member Data Documentation

24.160.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::](#)

[InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

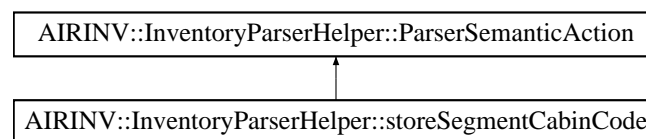
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.161 AIRINV::InventoryParserHelper::storeSegmentCabinCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentCabinCode:



Public Member Functions

- [storeSegmentCabinCode](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (char iChar) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.161.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 245 of file [InventoryParserHelper.hpp](#).

24.161.2 Constructor & Destructor Documentation

24.161.2.1 AIRINV::InventoryParserHelper::storeSegmentCabinCode::storeSegmentCabinCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 446 of file [InventoryParserHelper.cpp](#).

24.161.3 Member Function Documentation

24.161.3.1 void AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator() (char iChar) const

Actor Function (functor).

Definition at line 451 of file [InventoryParserHelper.cpp](#).

References [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::SegmentStruct::_cabinList](#), [AIRINV::BookingClassStruct::_classCode](#), [AIRINV::FareFamilyStruct::_classList](#), [AIRINV::SegmentCabinStruct::_fareFamilies](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), [AIRINV::SegmentCabinStruct::_itFareFamily](#), [AIRINV::FlightDateStruct::_itSegment](#), and [AIRINV::FlightDateStruct::_itSegmentCabin](#).

24.161.4 Member Data Documentation

24.161.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

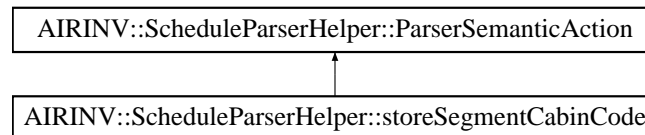
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.162 AIRINV::ScheduleParserHelper::storeSegmentCabinCode Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentCabinCode:



Public Member Functions

- [storeSegmentCabinCode](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.162.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 160 of file [ScheduleParserHelper.hpp](#).

24.162.2 Constructor & Destructor Documentation

24.162.2.1 AIRINV::ScheduleParserHelper::storeSegmentCabinCode::storeSegmentCabinCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 299 of file [ScheduleParserHelper.cpp](#).

24.162.3 Member Function Documentation

24.162.3.1 void AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 304 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::SegmentCabinStruct::_cabinCode](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRINV::FlightPeriodStruct::_itSegmentCabin](#).

24.162.4 Member Data Documentation

24.162.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod
[inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

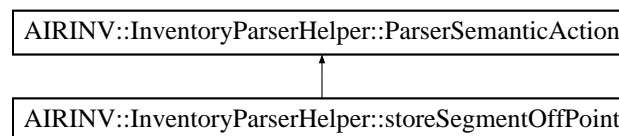
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.163 AIRINV::InventoryParserHelper::storeSegmentOffPoint Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentOffPoint:



Public Member Functions

- [storeSegmentOffPoint](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.163.1 Detailed Description

Store the parsed segment off point.

Definition at line 237 of file [InventoryParserHelper.hpp](#).

24.163.2 Constructor & Destructor Documentation

24.163.2.1 AIRINV::InventoryParserHelper::storeSegmentOffPoint::storeSegmentOffPoint ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 432 of file [InventoryParserHelper.cpp](#).

24.163.3 Member Function Documentation

24.163.3.1 void AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 437 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itSegment](#), and [AIRINV::SegmentStruct::_offPoint](#).

24.163.4 Member Data Documentation

24.163.4.1 **FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate**
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

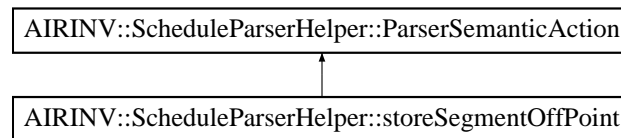
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.164 AIRINV::ScheduleParserHelper::storeSegmentOffPoint Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentOffPoint:



Public Member Functions

- [storeSegmentOffPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.164.1 Detailed Description

Store the parsed segment off point.

Definition at line 152 of file [ScheduleParserHelper.hpp](#).

24.164.2 Constructor & Destructor Documentation

24.164.2.1 AIRINV::ScheduleParserHelper::storeSegmentOffPoint::storeSegmentOffPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 286 of file [ScheduleParserHelper.cpp](#).

24.164.3 Member Function Documentation

24.164.3.1 void AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 291 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRINV::FlightPeriodStruct::_itSegment](#), and [AIRINV::SegmentStruct::_offPoint](#).

24.164.4 Member Data Documentation

24.164.4.1 [FlightPeriodStruct](#)& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRINV::ScheduleParserHelper::storeCapacity](#).

[::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

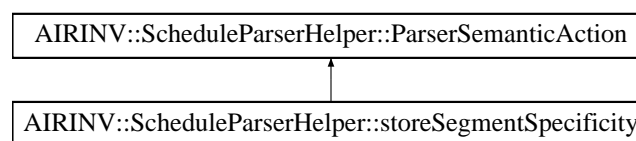
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.165 AIRINV::ScheduleParserHelper::storeSegmentSpecificity Struct Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentSpecificity:



Public Member Functions

- [storeSegmentSpecificity](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

24.165.1 Detailed Description

Store whether or not the segment definitions are specific. Specific means that there is a definition for each segment. General (not specific) means that a single definition defines all the segments.

Definition at line 136 of file [ScheduleParserHelper.hpp](#).

24.165.2 Constructor & Destructor Documentation

24.165.2.1 AIRINV::ScheduleParserHelper::storeSegmentSpecificity::storeSegmentSpecificity ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 247 of file [ScheduleParserHelper.cpp](#).

24.165.3 Member Function Documentation

24.165.3.1 void AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 252 of file [ScheduleParserHelper.cpp](#).

References [AIRINV::FlightPeriodStruct::_airportList](#), [AIRINV::FlightPeriodStruct::_airportOrderedList](#), [AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRINV::FlightPeriodStruct::buildSegments\(\)](#).

24.165.4 Member Data Documentation

24.165.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRINV::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeCapacity::operator\(\)](#), [operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::ScheduleParserHelper::doEndFlight::operator\(\)](#).

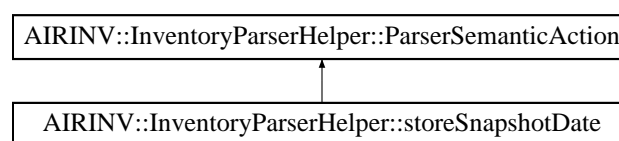
The documentation for this struct was generated from the following files:

- [airinv/command/ScheduleParserHelper.hpp](#)
- [airinv/command/ScheduleParserHelper.cpp](#)

24.166 AIRINV::InventoryParserHelper::storeSnapshotDate Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSnapshotDate:



Public Member Functions

- [storeSnapshotDate](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.166.1 Detailed Description

Store the snapshot date.

Definition at line 37 of file [InventoryParserHelper.hpp](#).

24.166.2 Constructor & Destructor Documentation

24.166.2.1 AIRINV::InventoryParserHelper::storeSnapshotDate::storeSnapshotDate (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 32 of file [InventoryParserHelper.cpp](#).

24.166.3 Member Function Documentation

24.166.3.1 void AIRINV::InventoryParserHelper::storeSnapshotDate::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 37 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_flightDate](#), and [AIRINV::FlightDateStruct::getDate\(\)](#).

24.166.4 Member Data Documentation

24.166.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFCClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

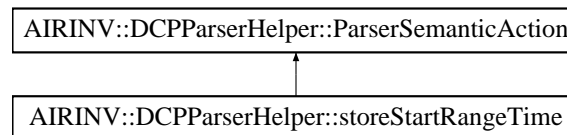
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.167 AIRINV::DCPParserHelper::storeStartRangeTime Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeStartRangeTime:



Public Member Functions

- [storeStartRangeTime](#) (DCPRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & [_DCPRule](#)

24.167.1 Detailed Description

Store the parsed start range time.

Definition at line 88 of file [DCPParserHelper.hpp](#).

24.167.2 Constructor & Destructor Documentation

24.167.2.1 AIRINV::DCPParserHelper::storeStartRangeTime::storeStartRangeTime (DCPRuleStruct & ioDCPRule)

Actor Constructor.

Definition at line 116 of file [DCPParserHelper.cpp](#).

24.167.3 Member Function Documentation

24.167.3.1 void AIRINV::DCPParserHelper::storeStartRangeTime::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 121 of file [DCPParserHelper.cpp](#).

References [AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule](#).

24.167.4 Member Data Documentation

24.167.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file [DCPParserHelper.hpp](#).

Referenced by [AIRINV::DCPParserHelper::storeDCPID::operator\(\)](#), [AIRINV::DCPParserHelper::storeOrigin::operator\(\)](#), [AIRINV::DCPParserHelper::storeDestination::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeStart::operator\(\)](#), [AIRINV::DCPParserHelper::storeDateRangeEnd::operator\(\)](#), [operator\(\)](#), [AIRINV::DCPParserHelper::storeEndRangeTime::operator\(\)](#), [AIRINV::DCPParserHelper::storePOS::operator\(\)](#), [AIRINV::DCPParserHelper::storeCabinCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeChannel::operator\(\)](#), [AIRINV::DCPParserHelper::storeAdvancePurchase::operator\(\)](#), [AIRINV::DCPParserHelper::storeSaturdayStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeChangeFees::operator\(\)](#), [AIRINV::DCPParserHelper::storeNonRefundable::operator\(\)](#), [AIRINV::DCPParserHelper::storeMinimumStay::operator\(\)](#), [AIRINV::DCPParserHelper::storeDCP::operator\(\)](#), [AIRINV::DCPParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::DCPParserHelper::storeClass::operator\(\)](#), and [AIRINV::DCPParserHelper::doEndDCP::operator\(\)](#).

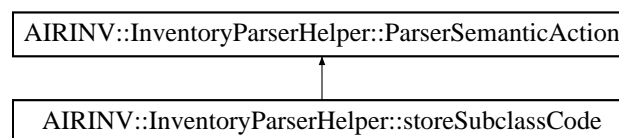
The documentation for this struct was generated from the following files:

- [airinv/command/vault/DCPParserHelper.hpp](#)
- [airinv/command/vault/DCPParserHelper.cpp](#)

24.168 AIRINV::InventoryParserHelper::storeSubclassCode Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeSubclassCode:



Public Member Functions

- [storeSubclassCode](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (unsigned int iNumber) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.168.1 Detailed Description

Store the parsed sub-class code.

Definition at line 269 of file [InventoryParserHelper.hpp](#).

24.168.2 Constructor & Destructor Documentation

24.168.2.1 AIRINV::InventoryParserHelper::storeSubclassCode::storeSubclassCode ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 511 of file [InventoryParserHelper.cpp](#).

24.168.3 Member Function Documentation

24.168.3.1 void AIRINV::InventoryParserHelper::storeSubclassCode::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 516 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBookingClass](#), and [AIRINV::BookingClassStruct::_subclassCode](#).

24.168.4 Member Data Documentation

24.168.4.1 **FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate**
[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeYieldUpperRange::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

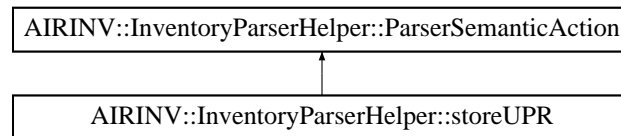
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.169 AIRINV::InventoryParserHelper::storeUPR Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeUPR:



Public Member Functions

- [storeUPR](#) ([FlightDateStruct](#) &)
- void [operator\(\)](#) (double *iReal*) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.169.1 Detailed Description

Store the parsed Unsold Protected (UPR).

Definition at line 157 of file [InventoryParserHelper.hpp](#).

24.169.2 Constructor & Destructor Documentation

24.169.2.1 AIRINV::InventoryParserHelper::storeUPR::storeUPR ([FlightDateStruct](#) & *ioFlightDate*)

Actor Constructor.

Definition at line 274 of file [InventoryParserHelper.cpp](#).

24.169.3 Member Function Documentation

24.169.3.1 void AIRINV::InventoryParserHelper::storeUPR::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 279 of file [InventoryParserHelper.cpp](#).

References [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::LegCabinStruct::_upr](#).

24.169.4 Member Data Documentation

24.169.4.1 [FlightDateStruct](#)& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

[inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParser](#)

Helper::storeSaleableCapacity::operator(), AIRINV::InventoryParserHelper::storeAU::operator(), operator(), AIRINV::InventoryParserHelper::storeBookingCounter::operator(), AIRINV::InventoryParserHelper::storeNAV::operator(), AIRINV::InventoryParserHelper::storeGAV::operator(), AIRINV::InventoryParserHelper::storeACP::operator(), AIRINV::InventoryParserHelper::storeETB::operator(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator(), AIRINV::InventoryParserHelper::storeBucketAvailability::operator(), AIRINV::InventoryParserHelper::storeSeatIndex::operator(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator(), AIRINV::InventoryParserHelper::storeClassCode::operator(), AIRINV::InventoryParserHelper::storeSubclassCode::operator(), AIRINV::InventoryParserHelper::storeParentClassCode::operator(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator(), AIRINV::InventoryParserHelper::storeProtection::operator(), AIRINV::InventoryParserHelper::storeNego::operator(), AIRINV::InventoryParserHelper::storeNoShow::operator(), AIRINV::InventoryParserHelper::storeOverbooking::operator(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator(), AIRINV::InventoryParserHelper::storeClassETB::operator(), AIRINV::InventoryParserHelper::storeClassAvailability::operator(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator(), AIRINV::InventoryParserHelper::storeFamilyCode::operator(), AIRINV::InventoryParserHelper::storeFClasses::operator(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator().

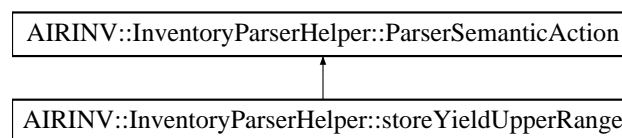
The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.170 AIRINV::InventoryParserHelper::storeYieldUpperRange Struct Reference

```
#include <airinv/command/InventoryParserHelper.hpp>
```

Inheritance diagram for AIRINV::InventoryParserHelper::storeYieldUpperRange:



Public Member Functions

- [storeYieldUpperRange](#) ([FlightDateStruct](#) &)
- [void operator\(\)](#) (double iReal) const

Public Attributes

- [FlightDateStruct](#) & [_flightDate](#)

24.170.1 Detailed Description

Store the parsed Yield Upper Range value.

Definition at line 205 of file [InventoryParserHelper.hpp](#).

24.170.2 Constructor & Destructor Documentation

24.170.2.1 AIRINV::InventoryParserHelper::storeYieldUpperRange::storeYieldUpperRange (FlightDateStruct & ioFlightDate)

Actor Constructor.

Definition at line 340 of file [InventoryParserHelper.cpp](#).

24.170.3 Member Function Documentation

24.170.3.1 void AIRINV::InventoryParserHelper::storeYieldUpperRange::operator() (double iReal) const

Actor Function (functor).

Definition at line 345 of file [InventoryParserHelper.cpp](#).

References [AIRINV::LegCabinStruct::_bucketList](#), [AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate](#), [AIRINV::FlightDateStruct::_itBucket](#), [AIRINV::FlightDateStruct::_itLegCabin](#), and [AIRINV::BucketStruct::_yieldRangeUpperValue](#).

24.170.4 Member Data Documentation

24.170.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file [InventoryParserHelper.hpp](#).

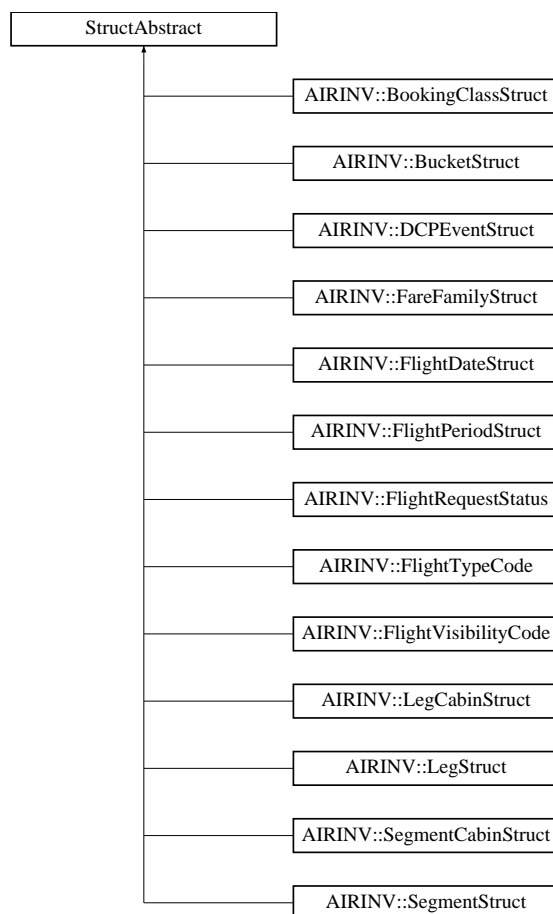
Referenced by [AIRINV::InventoryParserHelper::storeSnapshotDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAirlineCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightNumber::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightTypeCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBoardingTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffDate::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOffTime::operator\(\)](#), [AIRINV::InventoryParserHelper::storeLegCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSaleableCapacity::operator\(\)](#), [AIRINV::InventoryParserHelper::storeAU::operator\(\)](#), [AIRINV::InventoryParserHelper::storeUPR::operator\(\)](#), [AIRINV::InventoryParserHelper::storeBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeGAV::operator\(\)](#), [AIRINV::InventoryParserHelper::storeACP::operator\(\)](#), [AIRINV::InventoryParserHelper::storeETB::operator\(\)](#), [operator\(\)](#), [AIRINV::InventoryParserHelper::storeBucketAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSeatIndex::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentClassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeParentSubclassCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeCumulatedProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeProtection::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNego::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNoShow::operator\(\)](#), [AIRINV::InventoryParserHelper::storeOverbooking::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassETB::operator\(\)](#), [AIRINV::InventoryParserHelper::storeClassAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeSegmentAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeRevenueAvailability::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFamilyCode::operator\(\)](#), [AIRINV::InventoryParserHelper::storeFClasses::operator\(\)](#), and [AIRINV::InventoryParserHelper::doEndFlightDate::operator\(\)](#).

The documentation for this struct was generated from the following files:

- [airinv/command/InventoryParserHelper.hpp](#)
- [airinv/command/InventoryParserHelper.cpp](#)

24.171 StructAbstract Class Reference

Inheritance diagram for StructAbstract:

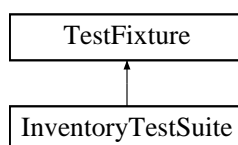


The documentation for this class was generated from the following file:

- [airinv/bom/SegmentStruct.hpp](#)

24.172 TestFixture Class Reference

Inheritance diagram for TestFixture:



The documentation for this class was generated from the following file:

- test/airinv/[InventoryTestSuite.hpp](#)

25 File Documentation

25.1 airinv/AIRINV_Master_Service.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <airrac/AIRAC_Types.hpp>
```

Classes

- class [AIRINV::AIRINV_Master_Service](#)
Interface for the [AIRINV](#) Services.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.2 AIRINV_Master_Service.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINV_MASTER_SERVICE_HPP
00002 #define __AIRINV_SVC_AIRINV_MASTER_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_service_types.hpp>
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/basic/ForecastingMethod.hpp>
00015 #include <stdair/basic/PartnershipTechnique.hpp>
00016 // AirRAC
00017 #include <airrac/AIRAC_Types.hpp>
00018
00019
00021 namespace stdair {
00022     class AirlineFeatureSet;
00023     class Inventory;
00024     class STDAIR_Service;
00025     struct BasLogParams;
00026     struct BasDBParams;
00027     struct SnapshotStruct;
00028     struct RMEventStruct;
00029     struct TravelSolutionStruct;
00030 }
00031
00032 namespace AIRINV {
00033
00035     class AIRINV_Master_ServiceContext;
00036
00037
00041     class AIRINV_Master_Service {
00042     public:
```

```

00043 // ////////// Constructors and destructors //////////
00059 AIRINV_Master_Service (const stdair::BasLogParams&,
00060                       const stdair::BasDBParams&);
00061
00073 AIRINV_Master_Service (const stdair::BasLogParams&);
00074
00090 AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T);
00091
00100 void parseAndLoad (const stdair::Filename_T& iInventoryFilename);
00101
00112 void parseAndLoad (const stdair::Filename_T& iScheduleFilename,
00113                  const stdair::Filename_T& iODInputFilename,
00114                  const AIRRAC::YieldFilePath& iYieldFilename);
00115
00119 ~AIRINV_Master_Service();
00120
00125 void initSnapshotAndRMEvents (const stdair::Date_T&, const stdair::Date_T&)
;
00126
00127 public:
00128 // ////////// Business Methods //////////
00137 void buildSampleBom();
00138
00142 void calculateAvailability (stdair::TravelSolutionStruct&,
00143                          const stdair::PartnershipTechnique&);
00144
00153 bool sell (const std::string& iSegmentDateKey, const stdair::ClassCode_T&,
00154          const stdair::PartySize_T&);
00164 bool cancel (const std::string& iSegmentDateKey, const stdair::ClassCode_T&
00165            const stdair::PartySize_T&);
00166
00170 void takeSnapshots (const stdair::SnapshotStruct&);
00171
00175 void optimise (const stdair::RMEventStruct&,
00176              const stdair::ForecastingMethod&,
00177              const stdair::PartnershipTechnique&);
00178
00179 public:
00181 // ////////// Export support methods //////////
00192 std::string jsonExport (const stdair::AirlineCode_T&,
00193                      const stdair::FlightNumber_T&,
00194                      const stdair::Date_T& iDepartureDate) const;
00195
00196 public:
00197 // ////////// Display support methods //////////
00212 std::string list (const stdair::AirlineCode_T& iAirlineCode = "all",
00213                const stdair::FlightNumber_T& iFlightNumber = 0) const;
00214
00224 bool check (const stdair::AirlineCode_T&, const stdair::FlightNumber_T&,
00225            const stdair::Date_T& iDepartureDate) const;
00226
00234 std::string csvDisplay() const;
00235
00247 std::string csvDisplay (const stdair::AirlineCode_T&,
00248                      const stdair::FlightNumber_T&,
00249                      const stdair::Date_T& iDepartureDate) const;
00250
00251 private:
00252 // ////////// Construction and Destruction helper methods //////////
00253 AIRINV_Master_Service();
00257
00262 AIRINV_Master_Service (const AIRINV_Master_Service&);
00263
00273 stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00274                                              const stdair::BasDBParams&);
00275
00284 stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
;
00285
00294 void addStdAirService (stdair::STDAIR_ServicePtr_T,
00295                      const bool iOwnStdairService);
00296
00301 void initServiceContext();
00302
00309 void initSlaveAirinvService();
00310
00314 void finalise();
00315
00316 private:
00317 // ////////// Service Context //////////
00318 AIRINV_Master_ServiceContext* _airinvMasterServiceContext;
00322

```

```

00323     };
00324 }
00325 #endif // __AIRINV_SVC_AIRINV_MASTER_SERVICE_HPP

```

25.3 airinv/AIRINV_Service.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <airrac/AIRRAC_Types.hpp>

```

Classes

- class [AIRINV::AIRINV_Service](#)
Interface for the [AIRINV](#) Services.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.4 AIRINV_Service.hpp

```

00001 #ifndef __AIRINV_SVC_AIRINV_SERVICE_HPP
00002 #define __AIRINV_SVC_AIRINV_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_service_types.hpp>
00012 #include <stdair/basic/ForecastingMethod.hpp>
00013 #include <stdair/basic/PartnershipTechnique.hpp>
00014 #include <stdair/bom/RMEventTypes.hpp>
00015 // AirRAC
00016 #include <airrac/AIRRAC_Types.hpp>
00017
00019 namespace stdair {
00020     class AirlineFeatureSet;
00021     class STDAIR_Service;
00022     class Inventory;
00023     struct TravelSolutionStruct;
00024     struct BasLogParams;
00025     struct BasDBParams;
00026 }
00027
00028 namespace AIRINV {
00029
00031     class AIRINV_ServiceContext;
00032
00033
00037     class AIRINV_Service {
00038     public:
00039         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00055         AIRINV_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);
00056
00068         AIRINV_Service (const stdair::BasLogParams&);
00069
00086         AIRINV_Service (stdair::STDAIR_ServicePtr_T);
00087
00096         void parseAndLoad (const stdair::Filename_T& iInventoryFilename);

```

```

00097
00108 void parseAndLoad (const stdair::Filename_T& iScheduleFilename,
00109                  const stdair::Filename_T& iODInputFilename,
00110                  const AIRRAC::YieldFilePath& iYieldFilename);
00111
00115 ~AIRINV_Service();
00116
00117 public:
00118 // //////////// Business Methods ////////////
00119 void buildSampleBom();
00127
00128 stdair::RMEEventList_T initRMEEvents (const stdair::Date_T& iStartDate,
00134                                     const stdair::Date_T& iEndDate);
00135
00136 void calculateAvailability (stdair::TravelSolutionStruct&,
00140                             const stdair::PartnershipTechnique&);
00141
00142 bool sell (const std::string& iSegmentDateKey, const stdair::ClassCode_T&,
00151           const stdair::PartySize_T&);
00152
00153 bool cancel (const std::string& iSegmentDateKey, const stdair::ClassCode_T&
00163             const stdair::PartySize_T&);
00164
00165 void takeSnapshots (const stdair::AirlineCode_T&,
00169                   const stdair::DateTime_T&);
00170
00171 void optimise (const stdair::AirlineCode_T&,
00175                const stdair::KeyDescription_T&,
00176                const stdair::DateTime_T&,
00177                const stdair::ForecastingMethod&,
00178                const stdair::PartnershipTechnique&);
00179
00180 public:
00181 // //////////// Export support methods ////////////
00182 std::string jsonExport (const stdair::AirlineCode_T&,
00193                        const stdair::FlightNumber_T&,
00194                        const stdair::Date_T& iDepartureDate) const;
00195
00196 public:
00197 // //////////// Display support methods ////////////
00198 std::string list (const stdair::AirlineCode_T& iAirlineCode = "all",
00212                  const stdair::FlightNumber_T& iFlightNumber = 0) const;
00213
00214 bool check (const stdair::AirlineCode_T&, const stdair::FlightNumber_T&,
00224            const stdair::Date_T& iDepartureDate) const;
00225
00226 std::string csvDisplay() const;
00234
00235 std::string csvDisplay (const stdair::AirlineCode_T&,
00247                        const stdair::FlightNumber_T&,
00248                        const stdair::Date_T& iDepartureDate) const;
00249
00250 private:
00251 // //////////// Construction and Destruction helper methods ////////////
00252 AIRINV_Service ();
00253
00254 AIRINV_Service (const AIRINV_Service&);
00255
00256 stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00273                                                const stdair::BasDBParams&);
00274
00275 stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
00284 ;
00285
00286 void initRMOLService();
00289
00290 void initAIRRACService();
00294
00295 void addStdAirService (stdair::STDAIR_ServicePtr_T,
00304                       const bool iOwnStdairService);
00305
00306 void initServiceContext();
00311
00312 void initAirinvService();
00319
00320 void finalise();
00324
00325 private:
00326 // //////////// Service Context ////////////
00327 AIRINV_ServiceContext* _airinvServiceContext;
00332
00333 };
00334
00335 #endif // __AIRINV_SVC_AIRINV_SERVICE_HPP

```

25.5 airinv/AIRINV_Types.hpp File Reference

```
#include <map>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Classes

- class [AIRINV::InventoryFileParsingFailedException](#)
- class [AIRINV::ScheduleFileParsingFailedException](#)
- class [AIRINV::SegmentDateNotFoundException](#)
- class [AIRINV::InventoryInputFileNotFoundException](#)
- class [AIRINV::ScheduleInputFileNotFoundException](#)
- class [AIRINV::FlightDateDuplicationException](#)
- class [AIRINV::BookingException](#)

Namespaces

- namespace [AIRINV](#)

Typedefs

- typedef boost::shared_ptr
< AIRINV_Service > [AIRINV::AIRINV_ServicePtr_T](#)
- typedef boost::shared_ptr
< AIRINV_Master_Service > [AIRINV::AIRINV_Master_ServicePtr_T](#)
- typedef std::map< const
stdair::AirlineCode_T,
AIRINV_ServicePtr_T > [AIRINV::AIRINV_ServicePtr_Map_T](#)
- typedef std::map< const
stdair::DTD_T, double > [AIRINV::FRAT5Curve_T](#)

25.6 AIRINV_Types.hpp

```
00001 #ifndef __AIRINV_AIRINV_TYPES_HPP
00002 #define __AIRINV_AIRINV_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <map>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014
00015 namespace AIRINV {
00016
00017 // Forward declarations
00018 class AIRINV_Service;
00019 class AIRINV_Master_Service;
00020
00021
00022 // /////////// Exceptions ///////////
00023
00027 class InventoryFileParsingFailedException
00028 : public stdair::ParsingFileFailedException {
00029 public:
00033 InventoryFileParsingFailedException (const std::string& iWhat)
00034 : stdair::ParsingFileFailedException (iWhat) {}
```

```

00035     };
00036
00040     class ScheduleFileParsingFailedException
00041     : public stdair::ParsingFileFailedException {
00042     public:
00046         ScheduleFileParsingFailedException (const std::string& iWhat)
00047         : stdair::ParsingFileFailedException (iWhat) {}
00048     };
00049
00054     class SegmentDateNotFoundException : public stdair::ParserException {
00055     public:
00059         SegmentDateNotFoundException (const std::string& iWhat)
00060         : stdair::ParserException (iWhat) {}
00061     };
00062
00066     class InventoryInputFileNotFoundException : public
stdair::FileNotFoundException {
00067     public:
00071         InventoryInputFileNotFoundException (const std::string& iWhat)
00072         : stdair::FileNotFoundException (iWhat) {}
00073     };
00074
00078     class ScheduleInputFileNotFoundException : public
stdair::FileNotFoundException {
00079     public:
00083         ScheduleInputFileNotFoundException (const std::string& iWhat)
00084         : stdair::FileNotFoundException (iWhat) {}
00085     };
00086
00090     class FlightDateDuplicationException : public
stdair::ObjectCreationDuplicationException {
00091     public:
00095         FlightDateDuplicationException (const std::string& iWhat)
00096         : stdair::ObjectCreationDuplicationException (iWhat) {}
00097     };
00098
00102     class BookingException : public stdair::RootException {
00103     };
00104
00105
00106     // ////////// Type definitions //////////
00110     typedef boost::shared_ptr<AIRINV_Service> AIRINV_ServicePtr_T;
00111
00115     typedef boost::shared_ptr<AIRINV_Master_Service> AIRINV_Master_ServicePtr_T;
00116
00121     typedef std::map<const stdair::AirlineCode_T,
00122                     AIRINV_ServicePtr_T> AIRINV_ServicePtr_Map_T;
00123
00127     typedef std::map<const stdair::DTD_T, double> FRAT5Curve_T;
00128
00129 }
00130 #endif // __AIRINV_AIRINV_TYPES_HPP
00131

```

25.7 airinv/basic/BasConst.cpp File Reference

```

#include <airinv/basic/BasConst_General.hpp>
#include <airinv/basic/BasConst_Curves.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>

```

Namespaces

- namespace [AIRINV](#)

Variables

- const std::string [AIRINV::DEFAULT_AIRLINE_CODE](#) = "BA"
- const [FRAT5Curve_T](#) [AIRINV::DEFAULT_PICKUP_FRAT5_CURVE](#)

25.8 BasConst.cpp

```

00001 // //////////////////////////////////////

```

```

00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 #include <airinv/basic/BasConst_General.hpp>
00005 #include <airinv/basic/BasConst_Curves.hpp>
00006 #include <airinv/basic/BasConst_AIRINV_Service.hpp>
00007
00008 namespace AIRINV {
00009
00011     const std::string DEFAULT_AIRLINE_CODE = "BA";
00012
00014     const FRAT5Curve_T DEFAULT_PICKUP_FRAT5_CURVE =
00015         DefaultMap::createPickupFRAT5Curve();
00016     FRAT5Curve_T DefaultMap::createPickupFRAT5Curve() {
00017         FRAT5Curve_T oCurve;
00018         // oCurve[365] = 1.1; oCurve[63] = 1.4; oCurve[56] = 1.45;
00019         // oCurve[49] = 1.5; oCurve[42] = 1.55; oCurve[35] = 1.6;
00020         // oCurve[31] = 1.7; oCurve[27] = 1.8; oCurve[23] = 2.0;
00021         // oCurve[19] = 2.3; oCurve[16] = 2.6; oCurve[13] = 3.0;
00022         // oCurve[10] = 3.3; oCurve[7] = 3.4; oCurve[5] = 3.44;
00023         // oCurve[3] = 3.47; oCurve[1] = 3.5; oCurve[0] = 3.5;
00024         // oCurve[365] = 1.0; oCurve[63] = 1.1; oCurve[56] = 1.13;
00025         // oCurve[49] = 1.17; oCurve[42] = 1.22; oCurve[35] = 1.28;
00026         // oCurve[31] = 1.32; oCurve[27] = 1.37; oCurve[23] = 1.43;
00027         // oCurve[19] = 1.51; oCurve[16] = 1.6; oCurve[13] = 1.7;
00028         // oCurve[10] = 1.8; oCurve[7] = 1.9; oCurve[5] = 1.93;
00029         // oCurve[3] = 1.96; oCurve[1] = 2.0; oCurve[0] = 2.0;
00030         // oCurve[365] = 1.0; oCurve[63] = 1.05; oCurve[56] = 1.07;
00031         // oCurve[49] = 1.09; oCurve[42] = 1.11; oCurve[35] = 1.14;
00032         // oCurve[31] = 1.16; oCurve[27] = 1.18; oCurve[23] = 1.21;
00033         // oCurve[19] = 1.24; oCurve[16] = 1.27; oCurve[13] = 1.3;
00034         // oCurve[10] = 1.33; oCurve[7] = 1.37; oCurve[5] = 1.4;
00035         // oCurve[3] = 1.45; oCurve[1] = 1.5; oCurve[0] = 1.5;
00036         // oCurve[365] = 1.1; oCurve[63] = 1.4;
00037         // oCurve[49] = 1.5; oCurve[35] = 1.6;
00038         // oCurve[23] = 2.0; oCurve[16] = 2.6;
00039         // oCurve[10] = 3.3; oCurve[5] = 3.44;
00040         // oCurve[1] = 3.5; oCurve[0] = 3.5;
00041         // oCurve[365] = 1.1; oCurve[63] = 1.4;
00042         // oCurve[49] = 1.7; oCurve[48] = 3.6; oCurve[35] = 3.6; oCurve[24] = 3.6;
00043         // oCurve[23] = 2.6; oCurve[16] = 2.7;
00044         // oCurve[10] = 3.2; oCurve[5] = 3.24; oCurve[4] = 2.8;
00045         // oCurve[1] = 2.4; oCurve[0] = 2.4;
00046
00047         oCurve[365] = 1.1; oCurve[63] = 1.4;
00048         /*1*/oCurve[62] = 1.4; oCurve[56] = 1.45;
00049         /*2*/oCurve[55] = 1.45; oCurve[49] = 1.5;
00050         /*3*/oCurve[48] = 1.5; oCurve[42] = 1.55;
00051         /*4*/oCurve[41] = 1.95; oCurve[35] = 2.2;
00052         /*5*/oCurve[34] = 2.2; oCurve[31] = 2.4;
00053         /*6*/oCurve[30] = 2.4; oCurve[27] = 2.8;
00054         /*7*/oCurve[26] = 2.9; oCurve[23] = 3.1;
00055         /*8*/oCurve[22] = 3.1; oCurve[19] = 3.3;
00056         /*9*/oCurve[18] = 3.3; oCurve[16] = 3.3;
00057         /*10*/oCurve[15] = 3.3; oCurve[13] = 3.3;
00058         /*11*/oCurve[12] = 3.0; oCurve[10] = 3.1;
00059         /*12*/oCurve[9] = 3.1; oCurve[7] = 3.1;
00060         /*13*/oCurve[6] = 3.1; oCurve[5] = 3.0;
00061         /*14*/oCurve[4] = 3.1; oCurve[3] = 3.0;
00062         /*15*/oCurve[2] = 3.0; oCurve[1] = 2.8;
00063         /*16*/oCurve[0] = 2.8;
00064
00065
00066         // oCurve[365] = 1.1; oCurve[63] = 1.4;
00067         // /*1*/oCurve[62] = 1.4; oCurve[56] = 1.55;
00068         // /*2*/oCurve[55] = 1.55; oCurve[49] = 1.7;
00069         // /*3*/oCurve[48] = 3.6; oCurve[42] = 3.6;
00070         // /*4*/oCurve[41] = 3.6; oCurve[35] = 3.6;
00071         // /*5*/oCurve[34] = 3.6; oCurve[31] = 3.6;
00072         // /*6*/oCurve[30] = 3.6; oCurve[27] = 3.6;
00073         // /*7*/oCurve[26] = 3.6; oCurve[23] = 3.6;
00074         // /*8*/oCurve[22] = 3.5; oCurve[19] = 3.3;
00075         // /*9*/oCurve[18] = 3.3; oCurve[16] = 3.0;
00076         // /*10*/oCurve[15] = 2.8; oCurve[13] = 2.5;
00077         // /*11*/oCurve[12] = 2.9; oCurve[10] = 3.2;
00078         // /*12*/oCurve[9] = 3.2; oCurve[7] = 3.22;
00079         // /*13*/oCurve[6] = 3.25; oCurve[5] = 3.3;
00080         // /*14*/oCurve[4] = 3.0; oCurve[3] = 2.8;
00081         // /*15*/oCurve[2] = 2.7; oCurve[1] = 2.5;
00082         // /*16*/oCurve[0] = 2.5;
00083
00084
00085         // oCurve[365] = 1.1; oCurve[63] = 1.4;
00086         // /*1*/oCurve[62] = 1.4; oCurve[49] = 1.7;
00087         // /*2*/oCurve[48] = 3.6; oCurve[35] = 3.6;
00088         // /*3*/oCurve[34] = 3.5; oCurve[23] = 3.4;
00089         // /*4*/oCurve[22] = 3.3; oCurve[16] = 3.1;
00090         // /*5*/oCurve[15] = 2.7; oCurve[10] = 3.1;

```

```

00091     // /*6*/oCurve[9] = 3.0;  oCurve[5]  = 2.8;
00092     // /*7*/oCurve[4] = 2.3;  oCurve[1]  = 2.5;
00093     // /*8*/oCurve[0]  = 2.5;
00094     return oCurve;
00095 };
00096
00097 }

```

25.9 airinv/basic/BasConst_AIRINV_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace [AIRINV](#)

25.10 BasConst_AIRINV_Service.hpp

```

00001 #ifndef __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP
00002 #define __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 #include <string>
00008
00009 namespace AIRINV {
00010
00012     extern const std::string DEFAULT_AIRLINE_CODE;
00013
00014 }
00015 #endif // __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP

```

25.11 airinv/basic/BasConst_Curves.hpp File Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Classes

- struct [AIRINV::DefaultMap](#)

Namespaces

- namespace [AIRINV](#)

25.12 BasConst_Curves.hpp

```

00001 #ifndef __AIRINV_BAS_BASCONST_CURVES_HPP
00002 #define __AIRINV_BAS_BASCONST_CURVES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // AIRINV
00008 #include <airinv/AIRINV_Types.hpp>
00009
00010 namespace AIRINV {
00011
00013     extern const FRAT5Curve_T DEFAULT_PICKUP_FRAT5_CURVE;
00014
00016     struct DefaultMap {
00017         static FRAT5Curve_T createPickupFRAT5Curve();

```



```

00018     };
00019 }
00020 #endif // __AIRINV_BAS_BASCONST_CURVES_HPP

```

25.13 airinv/basic/BasConst_General.hpp File Reference

Namespaces

- namespace [AIRINV](#)

25.14 BasConst_General.hpp

```

00001 #ifndef __AIRINV_BAS_BASCONST_GENERAL_HPP
00002 #define __AIRINV_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace AIRINV {
00009
00010 }
00011 #endif // __AIRINV_BAS_BASCONST_GENERAL_HPP

```

25.15 airinv/basic/BasParserTypes.hpp File Reference

```

#include <string>
#include <boost/spirit/home/classic/core.hpp>
#include <boost/spirit/home/classic/attribute.hpp>
#include <boost/spirit/home/classic/utility/functor_parser.hpp>
#include <boost/spirit/home/classic/utility/loops.hpp>
#include <boost/spirit/home/classic/utility/chset.hpp>
#include <boost/spirit/home/classic/utility/confix.hpp>
#include <boost/spirit/home/classic/iterator/file_iterator.hpp>
#include <boost/spirit/home/classic/actor/push_back_actor.hpp>
#include <boost/spirit/home/classic/actor/assign_actor.hpp>

```

Namespaces

- namespace [AIRINV](#)

Typedefs

- typedef char [AIRINV::char_t](#)
- typedef
boost::spirit::classic::file_iterator
< char_t > [AIRINV::iterator_t](#)
- typedef
boost::spirit::classic::scanner
< iterator_t > [AIRINV::scanner_t](#)
- typedef
boost::spirit::classic::rule
< scanner_t > [AIRINV::rule_t](#)
- typedef
boost::spirit::classic::int_parser
< unsigned int, 10, 1, 1 > [AIRINV::int1_p_t](#)

- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 2, 2 > [AIRINV::uint2_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 2 > [AIRINV::uint1_2_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 3 > [AIRINV::uint1_3_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 4, 4 > [AIRINV::uint4_p_t](#)
- typedef
boost::spirit::classic::uint_parser
< unsigned int, 10, 1, 4 > [AIRINV::uint1_4_p_t](#)
- typedef
boost::spirit::classic::chset
< char_t > [AIRINV::chset_t](#)
- typedef
boost::spirit::classic::impl::loop_traits
< chset_t, unsigned int,
unsigned int >::type [AIRINV::repeat_p_t](#)
- typedef
boost::spirit::classic::bounded
< uint2_p_t, unsigned int > [AIRINV::bounded2_p_t](#)
- typedef
boost::spirit::classic::bounded
< uint1_2_p_t, unsigned int > [AIRINV::bounded1_2_p_t](#)
- typedef
boost::spirit::classic::bounded
< uint1_3_p_t, unsigned int > [AIRINV::bounded1_3_p_t](#)
- typedef
boost::spirit::classic::bounded
< uint4_p_t, unsigned int > [AIRINV::bounded4_p_t](#)
- typedef
boost::spirit::classic::bounded
< uint1_4_p_t, unsigned int > [AIRINV::bounded1_4_p_t](#)

25.16 BasParserTypes.hpp

```

00001 #ifndef __AIRINV_BAS_BASCOMPARSERTYPES_HPP
00002 #define __AIRINV_BAS_BASCOMPARSERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 // #define BOOST_SPIRIT_DEBUG
00011 #include <boost/spirit/home/classic/core.hpp>
00012 #include <boost/spirit/home/classic/attribute.hpp>
00013 #include <boost/spirit/home/classic/utility/functor_parser.hpp>
00014 #include <boost/spirit/home/classic/utility/loops.hpp>
00015 #include <boost/spirit/home/classic/utility/chset.hpp>
00016 #include <boost/spirit/home/classic/utility/config.hpp>
00017 #include <boost/spirit/home/classic/iterator/file_iterator.hpp>
00018 #include <boost/spirit/home/classic/actor/push_back_actor.hpp>
00019 #include <boost/spirit/home/classic/actor/assign_actor.hpp>
00020
00021 namespace AIRINV {
00022
00023 // //////////////////////////////////////
00024 //
00025 // Definition of Basic Types

```

```

00026 //
00027 // //////////////////////////////////////
00028 // For a file, the parsing unit is the character (char). For a string,
00029 // it is a "char const *".
00030 // typedef char const* iterator_t;
00031 typedef char char_t;
00032
00033 // The types of iterator, scanner and rule are then derived from
00034 // the parsing unit.
00035 typedef boost::spirit::classic::file_iterator<char_t> iterator_t;
00036 typedef boost::spirit::classic::scanner<iterator_t> scanner_t;
00037 typedef boost::spirit::classic::rule<scanner_t> rule_t;
00038
00039 // //////////////////////////////////////
00040 //
00041 // Parser related types
00042 //
00043 // //////////////////////////////////////
00045 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00046
00048 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> uint2_p_t
;
00049
00051 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2>
uint1_2_p_t;
00052
00054 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3>
uint1_3_p_t;
00055
00057 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t
;
00058
00060 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
uint1_4_p_t;
00061
00063 typedef boost::spirit::classic::chset<char_t> chset_t;
00064
00067 typedef boost::spirit::classic::impl::loop_traits<chset_t,
00068 unsigned int,
00069 unsigned int>::type repeat_p_t;
00070
00072 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t
;
00073
00075 typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int>
bounded1_2_p_t;
00074 typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int>
bounded1_3_p_t;
00075 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t
;
00076 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
bounded1_4_p_t;
00077 }
00078 #endif // __AIRINV_BAS_BASCOMPARSERTYPES_HPP

```

25.17 airinv/basic/FlightRequestStatus.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/FlightRequestStatus.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.18 FlightRequestStatus.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```

```

00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // AirInv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/FlightRequestStatus.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 const std::string FlightRequestStatus::_labels[LAST_VALUE] =
00017     { "OK", "Not Found", "Internal Error" };
00018
00019 const std::string FlightRequestStatus::_codeLabels[LAST_VALUE] =
00020     { "OK", "NF", "IE" };
00021
00022 // //////////////////////////////////////
00023 FlightRequestStatus::
00024 FlightRequestStatus (const EN_FlightRequestStatus& iFlightRequestStatus)
00025     : _code (iFlightRequestStatus) {
00026 }
00027
00028 // //////////////////////////////////////
00029 FlightRequestStatus::FlightRequestStatus (const std::string& iCode) {
00030     _code = LAST_VALUE;
00031
00032     if (iCode == "OK") {
00033         _code = OK;
00034
00035     } else if (iCode == "NF") {
00036         _code = NOT_FOUND;
00037
00038     } else if (iCode == "IE") {
00039         _code = INTERNAL_ERROR;
00040
00041     }
00042
00043     if (_code == LAST_VALUE) {
00044         const std::string& lLabels = describeLabels();
00045         STDAIR_LOG_ERROR ("The flight request status '" << iCode
00046             << "' is not known. Known flight request status: "
00047             << lLabels);
00048         throw stdair::CodeConversionException ("The flight request status '"
00049             + iCode
00050             + "' is not known. Known flight
00051 request status: "
00052             + lLabels);
00053     }
00054 }
00055
00056 // //////////////////////////////////////
00057 const std::string& FlightRequestStatus::
00058 getLabel (const EN_FlightRequestStatus& iCode) {
00059     return _labels[iCode];
00060 }
00061
00062 // //////////////////////////////////////
00063 const std::string& FlightRequestStatus::
00064 getCodeLabel (const EN_FlightRequestStatus& iCode) {
00065     return _codeLabels[iCode];
00066 }
00067
00068 // //////////////////////////////////////
00069 std::string FlightRequestStatus::describeLabels() {
00070     std::ostringstream ostr;
00071     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00072         if (idx != 0) {
00073             ostr << ", ";
00074         }
00075         ostr << _labels[idx];
00076     }
00077     return ostr.str();
00078 }
00079
00080 // //////////////////////////////////////
00081 FlightRequestStatus::EN_FlightRequestStatus FlightRequestStatus::
00082 getCode() const {
00083     return _code;
00084 }
00085
00086 // //////////////////////////////////////
00087 const std::string FlightRequestStatus::describe() const {
00088     std::ostringstream ostr;
00089     ostr << _labels[_code];
00090     return ostr.str();
00091 }
00092

```

```
00093 }
```

25.19 airinv/basic/FlightTypeCode.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/FlightTypeCode.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.20 FlightTypeCode.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/basic/FlightTypeCode.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 const std::string FlightTypeCode::_labels[LAST_VALUE] =
00017     { "Domestic", "International", "Ground Handling" };
00018
00019 const std::string FlightTypeCode::_codeLabels[LAST_VALUE] =
00020     { "DOM", "INT", "GRD" };
00021
00022
00023 // //////////////////////////////////////
00024 FlightTypeCode::FlightTypeCode (const EN_FlightTypeCode& iFlightTypeCode)
00025     : _code (iFlightTypeCode) {
00026 }
00027
00028 // //////////////////////////////////////
00029 FlightTypeCode::FlightTypeCode (const std::string& iCode) {
00030     _code = LAST_VALUE;
00031
00032     if (iCode == "DOM") {
00033         _code = DOMESTIC;
00034     } else if (iCode == "INT") {
00035         _code = INTERNATIONAL;
00036     } else if (iCode == "GRD") {
00037         _code = GROUND_HANDLING;
00038     }
00039
00040     if (_code == LAST_VALUE) {
00041         const std::string& lLabels = describeLabels();
00042         STDAIR_LOG_ERROR ("The flight type code '" << iCode
00043             << "' is not known. Known flight type codes: "
00044             << lLabels);
00045         throw stdair::CodeConversionException ("The flight type code '" + iCode
00046             + "' is not known. Known flight
00047             type codes: "
00048                 + lLabels);
00049     }
00050 }
00051
00052 // //////////////////////////////////////
00053 const std::string& FlightTypeCode::getLabel (const EN_FlightTypeCode& iCode)
00054 {
00055     return _labels[iCode];
00056 }
```

```

00057
00058 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00059 const std::string& FlightTypeCode::
00060 getCodeLabel (const EN_FlightTypeCode& iCode) {
00061     return _codeLabels[iCode];
00062 }
00063
00064 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00065 std::string FlightTypeCode::describeLabels() {
00066     std::ostringstream ostr;
00067     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00068         if (idx != 0) {
00069             ostr << ", ";
00070         }
00071         ostr << _labels[idx];
00072     }
00073     return ostr.str();
00074 }
00075
00076 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00077 FlightTypeCode::EN_FlightTypeCode FlightTypeCode::getCode() const {
00078     return _code;
00079 }
00080
00081 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00082 const std::string FlightTypeCode::describe() const {
00083     std::ostringstream ostr;
00084     ostr << _labels[_code];
00085     return ostr.str();
00086 }
00087
00088 }

```

25.21 airinv/basic/FlightTypeCode.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [AIRINV::FlightTypeCode](#)

Namespaces

- namespace [AIRINV](#)

25.22 FlightTypeCode.hpp

```

00001 #ifndef __AIRINV_BAS_FLIGHTTYPECODE_HPP
00002 #define __AIRINV_BAS_FLIGHTTYPECODE_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00014     struct FlightTypeCode : public stdair::StructAbstract {
00015     public:
00016         typedef enum {
00017             DOMESTIC = 0,
00018             INTERNATIONAL,
00019             GROUND_HANDLING,
00020             LAST_VALUE
00021         } EN_FlightTypeCode;
00022
00023         static const std::string& getLabel (const EN_FlightTypeCode&);
00024
00025         static const std::string& getCodeLabel (const EN_FlightTypeCode&);
00026
00027     };
00028 }

```

```

00029
00031     static std::string describeLabels();
00032
00034     EN_FlightTypeCode getCode() const;
00035
00037     const std::string describe() const;
00038
00039
00040 public:
00042     FlightTypeCode (const EN_FlightTypeCode&);
00044     FlightTypeCode (const std::string& iCode);
00045
00046
00047 private:
00049     static const std::string _labels[LAST_VALUE];
00051     static const std::string _codeLabels[LAST_VALUE];
00052
00053
00054 private:
00055     // ////////// Attributes //////////
00057     EN_FlightTypeCode _code;
00058 };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTTYPECODE_HPP

```

25.23 airinv/basic/FlightVisibilityCode.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/FlightVisibilityCode.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.24 FlightVisibilityCode.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/basic/FlightVisibilityCode.hpp>
00012
00013 namespace AIRINV {
00014
00015     // //////////////////////////////////////
00016     const std::string FlightVisibilityCode::_labels[LAST_VALUE] =
00017         { "Normal", "Hidden", "Pseudo" };
00018
00019     const std::string FlightVisibilityCode::_codeLabels[LAST_VALUE] =
00020         { "NOR", "HID", "PSD" };
00021
00022
00023     // //////////////////////////////////////
00024     FlightVisibilityCode:
00025     FlightVisibilityCode (const EN_FlightVisibilityCode& iFlightVisibilityCode)
00026         : _code (iFlightVisibilityCode) {
00027     }
00028
00029     // //////////////////////////////////////
00030     FlightVisibilityCode:FlightVisibilityCode (const std::string& iCode) {
00031         _code = LAST_VALUE;
00032
00033         if (iCode == "NOR") {
00034             _code = NORMAL;

```

```

00035
00036     } else if (iCode == "HID") {
00037         _code = HIDDEN;
00038
00039     } else if (iCode == "PSD") {
00040         _code = PSEUDO;
00041     }
00042
00043     if (_code == LAST_VALUE) {
00044         const std::string& lLabels = describeLabels();
00045         STDAIR_LOG_ERROR ("The flight visibility code '" << iCode
00046             << "' is not known. Known flight visibility codes: "
00047             << lLabels);
00048         throw stdair::CodeConversionException ("The flight visibility code '"
00049             + iCode
00050             + "' is not known. Known flight
visibility codes: "
00051             + lLabels);
00052     }
00053 }
00054
00055 // //////////////////////////////////////
00056 const std::string& FlightVisibilityCode::
00057 getLabel (const EN_FlightVisibilityCode& iCode) {
00058     return _labels[iCode];
00059 }
00060
00061 // //////////////////////////////////////
00062 const std::string& FlightVisibilityCode::
00063 getCodeLabel (const EN_FlightVisibilityCode& iCode) {
00064     return _codeLabels[iCode];
00065 }
00066
00067 // //////////////////////////////////////
00068 std::string FlightVisibilityCode::describeLabels() {
00069     std::ostringstream ostr;
00070     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00071         if (idx != 0) {
00072             ostr << ", ";
00073         }
00074         ostr << _labels[idx];
00075     }
00076     return ostr.str();
00077 }
00078
00079 // //////////////////////////////////////
00080 FlightVisibilityCode::EN_FlightVisibilityCode FlightVisibilityCode::
00081 getCode() const {
00082     return _code;
00083 }
00084
00085 // //////////////////////////////////////
00086 const std::string FlightVisibilityCode::describe() const {
00087     std::ostringstream ostr;
00088     ostr << _labels[_code];
00089     return ostr.str();
00090 }
00091
00092 }

```

25.25 airinv/basic/FlightVisibilityCode.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [AIRINV::FlightVisibilityCode](#)

Namespaces

- namespace [AIRINV](#)

25.26 FlightVisibilityCode.hpp

```

00001 #ifndef __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP
00002 #define __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015     struct FlightVisibilityCode : public stdair::StructAbstract {
00016     public:
00017         typedef enum {
00018             NORMAL = 0,
00019             HIDDEN,
00020             PSEUDO,
00021             LAST_VALUE
00022         } EN_FlightVisibilityCode;
00023
00025         static const std::string& getLabel (const EN_FlightVisibilityCode&);
00026
00028         static const std::string& getCodeLabel (const EN_FlightVisibilityCode&);
00029
00031         static std::string describeLabels();
00032
00034         EN_FlightVisibilityCode getCode() const;
00035
00037         const std::string describe() const;
00038
00039     public:
00042         FlightVisibilityCode (const EN_FlightVisibilityCode&);
00044         FlightVisibilityCode (const std::string& iCode);
00045
00046     private:
00049         static const std::string _labels[LAST_VALUE];
00051         static const std::string _codeLabels[LAST_VALUE];
00052
00053     private:
00055         // ////////// Attributes //////////
00057         EN_FlightVisibilityCode _code;
00058     };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP

```

25.27 airinv/batches/airinv_parseInventory.cpp File Reference

25.28 airinv_parseInventory.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 // StdAir
00015 #include <stdair/basic/BasLogParams.hpp>
00016 #include <stdair/basic/BasDBParams.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirInv
00019 #include <airinv/AIRINV_Master_Service.hpp>
00020 #include <airinv/config/airinv-paths.hpp>
00021
00022 // ////////// Constants //////////
00026 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinv_parseInventory.log");
00027
00031 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00032                                                         "/invdump01.csv");
00036 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00037                                                         "/schedule01.csv");

```

```

00041 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00042             "/ond01.csv");
00043
00047 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00048             "/yieldstore01.csv");
00049
00053 const std::string K_AIRINV_DEFAULT_SEGMENT_DATE_KEY ("SV,5,2010-03-11,KBP,JFK")
00054 ;
00058 const stdair::ClassCode_T K_AIRINV_DEFAULT_CLASS_CODE ("Y");
00059
00063 const stdair::PartySize_T K_AIRINV_DEFAULT_PARTY_SIZE (2);
00064
00069 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00070
00075 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00076
00080 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00081
00082 // /////////// Parsing of Options & Configuration ///////////
00083 // A helper function to simplify the main part.
00084 template<class T> std::ostream& operator<< (std::ostream& os,
00085             const std::vector<T>& v) {
00086     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00087     return os;
00088 }
00089
00093 int readConfiguration (int argc, char* argv[],
00094             bool& ioIsBuiltin, bool& ioIsForSchedule,
00095             stdair::Filename_T& ioInventoryFilename,
00096             stdair::Filename_T& ioScheduleInputFilename,
00097             stdair::Filename_T& ioODInputFilename,
00098             stdair::Filename_T& ioYieldInputFilename,
00099             std::string& ioSegmentDateKey,
00100             stdair::ClassCode_T& ioClassCode,
00101             stdair::PartySize_T& ioPartySize,
00102             std::string& ioLogFilename) {
00103     // Default for the built-in input
00104     ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00105
00106     // Default for the inventory or schedule option
00107     ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00108
00109     // Declare a group of options that will be allowed only on command line
00110     boost::program_options::options_description generic ("Generic options");
00111     generic.add_options()
00112         ("prefix", "print installation prefix")
00113         ("version,v", "print version string")
00114         ("help,h", "produce help message");
00115
00116     // Declare a group of options that will be allowed both on command
00117     // line and in config file
00118
00119     boost::program_options::options_description config ("Configuration");
00120     config.add_options()
00121         ("builtin,b",
00122             "The sample BOM tree can be either built-in or parsed from an input file.
00123             That latter must then be given with the -i/--inventory or -s/--schedule option")
00124         ("for_schedule,f",
00125             "The BOM tree should be built from a schedule file (instead of from an
00126             inventory dump)")
00127         ("inventory,i",
00128             boost::program_options::value< std::string >(&ioInventoryFilename)->
00129             default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00130             "(CSV) input file for the inventory")
00131         ("schedule,s",
00132             boost::program_options::value< std::string >(&ioScheduleInputFilename)->
00133             default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00134             "(CSV) input file for the schedule")
00135         ("ond,o",
00136             boost::program_options::value< std::string >(&ioODInputFilename)->
00137             default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00138             "(CSV) input file for the O&D")
00139         ("yield,y",
00140             boost::program_options::value< std::string >(&ioYieldInputFilename)->
00141             default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00142             "(CSV) input file for the yield")
00143         ("segment_date_key,k",
00144             boost::program_options::value< std::string >(&ioSegmentDateKey)->
00145             default_value(K_AIRINV_DEFAULT_SEGMENT_DATE_KEY),
00146             "Segment-date key")
00147         ("class_code,c",
00148             boost::program_options::value< stdair::ClassCode_T >(&ioClassCode)->
00149             default_value(K_AIRINV_DEFAULT_CLASS_CODE),
00150             "Class code")
00151         ("party_size,p",
00152             boost::program_options::value< stdair::PartySize_T >(&ioPartySize)->

```

```

    default_value(K_AIRINV_DEFAULT_PARTY_SIZE),
00145     "Party size")
00146     ("log,l",
00147     boost::program_options::value< std::string >(&ioLogFilename)->
    default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00148     "Filename for the logs")
00149     ;
00150
00151     // Hidden options, will be allowed both on command line and
00152     // in config file, but will not be shown to the user.
00153     boost::program_options::options_description hidden ("Hidden options");
00154     hidden.add_options()
00155         ("copyright",
00156         boost::program_options::value< std::vector<std::string> >(),
00157         "Show the copyright (license)");
00158
00159     boost::program_options::options_description cmdline_options;
00160     cmdline_options.add(generic).add(config).add(hidden);
00161
00162     boost::program_options::options_description config_file_options;
00163     config_file_options.add(config).add(hidden);
00164     boost::program_options::options_description visible ("Allowed options");
00165     visible.add(generic).add(config);
00166
00167     boost::program_options::positional_options_description p;
00168     p.add ("copyright", -1);
00169
00170     boost::program_options::variables_map vm;
00171     boost::program_options::
00172         store (boost::program_options::command_line_parser (argc, argv).
00173             options (cmdline_options).positional(p).run(), vm);
00174
00175     std::ifstream ifs ("airinv.cfg");
00176     boost::program_options::store (parse_config_file (ifs, config_file_options),
00177         vm);
00178     boost::program_options::notify (vm);
00179
00180     if (vm.count ("help")) {
00181         std::cout << visible << std::endl;
00182         return K_AIRINV_EARLY_RETURN_STATUS;
00183     }
00184
00185     if (vm.count ("version")) {
00186         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00187         return K_AIRINV_EARLY_RETURN_STATUS;
00188     }
00189
00190     if (vm.count ("prefix")) {
00191         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00192         return K_AIRINV_EARLY_RETURN_STATUS;
00193     }
00194
00195     if (vm.count ("builtin")) {
00196         ioIsBuiltin = true;
00197     }
00198     const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00199     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00200
00201     if (vm.count ("for_schedule")) {
00202         ioIsForSchedule = true;
00203     }
00204     const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00205     std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00206         << std::endl;
00207
00208     if (ioIsBuiltin == false) {
00209
00210         if (ioIsForSchedule == false) {
00211             // The BOM tree should be built from parsing an inventory dump
00212             if (vm.count ("inventory")) {
00213                 ioInventoryFilename = vm["inventory"].as< std::string >();
00214                 std::cout << "Input inventory filename is: " << ioInventoryFilename
00215                     << std::endl;
00216             } else {
00217                 // The built-in option is not selected. However, no inventory dump
00218                 // file is specified
00219                 std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00220                     << " -f/--for_schedule and -s/--schedule options "
00221                     << "must be specified" << std::endl;
00222             }
00223         } else {
00224
00225             // The BOM tree should be built from parsing a schedule (and O&D) file
00226             if (vm.count ("schedule")) {
00227                 ioScheduleInputFilename = vm["schedule"].as< std::string >();
00228                 std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00229

```

```

00230         << std::endl;
00231
00232     } else {
00233         // The built-in option is not selected. However, no schedule file
00234         // is specified
00235         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00236         << " -f/--for_schedule and -s/--schedule options "
00237         << "must be specified" << std::endl;
00238     }
00239
00240     if (vm.count ("ond")) {
00241         ioODInputFilename = vm["ond"].as< std::string >();
00242         std::cout << "Input O&D filename is: " << ioODInputFilename <<
std::endl;
00243     }
00244
00245     if (vm.count ("yield")) {
00246         ioYieldInputFilename = vm["yield"].as< std::string >();
00247         std::cout << "Input yield filename is: "
00248         << ioYieldInputFilename << std::endl;
00249     }
00250 }
00251 }
00252
00253 if (vm.count ("log")) {
00254     ioLogFilename = vm["log"].as< std::string >();
00255     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00256 }
00257
00258 return 0;
00259 }
00260
00261
00262 // ////////// M A I N //////////
00263 int main (int argc, char* argv[]) {
00264
00265     // State whether the BOM tree should be built-in or parsed from an
00266     // input file
00267     bool isBuiltin;
00268     bool isForSchedule;
00269
00270     // Input file names
00271     stdair::Filename_T lInventoryFilename;
00272     stdair::Filename_T lScheduleInputFilename;
00273     stdair::Filename_T lODInputFilename;
00274     stdair::Filename_T lYieldInputFilename;
00275
00276     // Parameters for the sale
00277     std::string lSegmentDateKey;
00278     stdair::ClassCode_T lClassCode;
00279     stdair::PartySize_T lPartySize;
00280
00281     // Output log File
00282     stdair::Filename_T lLogFilename;
00283
00284     // Call the command-line option parser
00285     const int lOptionParserStatus =
00286         readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
,
00287         lScheduleInputFilename, lODInputFilename,
00288         lYieldInputFilename, lSegmentDateKey, lClassCode,
00289         lPartySize, lLogFilename);
00290
00291     if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00292         return 0;
00293     }
00294
00295     // Set the log parameters
00296     std::ofstream logOutputFile;
00297     // Open and clean the log outputfile
00298     logOutputFile.open (lLogFilename.c_str());
00299     logOutputFile.clear();
00300
00301     // Initialise the inventory service
00302     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00303     AIRINV::AIRINV_Master_Service airinvService (lLogParams);
00304
00305     // DEBUG
00306     STDAIR_LOG_DEBUG ("Welcome to AirInv");
00307
00308     // Check whether or not a (CSV) input file should be read
00309     if (isBuiltin == true) {
00310
00311         // Build the sample BOM tree for RMOL
00312         airinvService.buildSampleBom();
00313
00314         // Define a specific segment-date key for the sample BOM tree

```

```

00315     lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00316
00317 } else {
00318     if (isForSchedule == true) {
00319         // Build the BOM tree from parsing a schedule file (and O&D list)
00320         AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00321         airinvService.parseAndLoad (lScheduleInputFilename, lODInputFilename,
00322                                     lYieldFilePath);
00323
00324         if (lSegmentDateKey == K_AIRINV_DEFAULT_SEGMENT_DATE_KEY) {
00325             // Define a specific segment-date key for the schedule-based inventory
00326             lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00327         }
00328
00329     } else {
00330         // Build the BOM tree from parsing an inventory dump file
00331         airinvService.parseAndLoad (lInventoryFilename);
00332     }
00333 }
00334
00335 // Make a booking
00336 const bool isSellSuccessful =
00337     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00338
00339 // DEBUG
00340 STDAIR_LOG_DEBUG ("Sale ('" << lSegmentDateKey << "', " << lClassCode << ": "
00341                  << lPartySize << ") successful? " << isSellSuccessful);
00342
00343 // DEBUG: Display the whole BOM tree
00344 const std::string& lCSVDump = airinvService.csvDisplay();
00345 STDAIR_LOG_DEBUG (lCSVDump);
00346
00347 // Close the Log outputFile
00348 logOutputFile.close();
00349
00350 /*
00351 Note: as that program is not intended to be run on a server in
00352 production, it is better not to catch the exceptions. When it
00353 happens (that an exception is throwned), that way we get the
00354 call stack.
00355 */
00356
00357 return 0;
00358 }

```

25.29 airinv/batches/parseInventory.cpp File Reference

25.30 parseInventory.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 // StdAir
00015 #include <stdair/basic/BasLogParams.hpp>
00016 #include <stdair/basic/BasDBParams.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirInv
00019 #include <airinv/AIRINV_Master_Service.hpp>
00020 #include <airinv/config/airinv-paths.hpp>
00021
00022 // ////////// Constants //////////
00026 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("parseInventory.log");
00027
00031 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00032                                                         "/invdump01.csv");
00036 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00037                                                         "/schedule01.csv");
00041 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00042                                                    "/ond01.csv");
00043
00047 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00048                                                      "/yieldstore01.csv");
00049
00053 const std::string K_AIRINV_DEFAULT_SEGMENT_DATE_KEY ("SV,5,2010-03-11,KBP,JFK")
00054 ;

```

```

00058 const stdair::ClassCode_T K_AIRINV_DEFAULT_CLASS_CODE ("Y");
00059
00063 const stdair::PartySize_T K_AIRINV_DEFAULT_PARTY_SIZE (2);
00064
00069 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00070
00075 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00076
00080 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00081
00082 // ////////// Parsing of Options & Configuration //////////
00083 // A helper function to simplify the main part.
00084 template<class T> std::ostream& operator<< (std::ostream& os,
00085                                           const std::vector<T>& v) {
00086     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00087     return os;
00088 }
00089
00093 int readConfiguration (int argc, char* argv[],
00094                        bool& ioIsBuiltin, bool& ioIsForSchedule,
00095                        stdair::Filename_T& ioInventoryFilename,
00096                        stdair::Filename_T& ioScheduleInputFilename,
00097                        stdair::Filename_T& ioODInputFilename,
00098                        stdair::Filename_T& ioYieldInputFilename,
00099                        std::string& ioSegmentDateKey,
00100                        stdair::ClassCode_T& ioClassCode,
00101                        stdair::PartySize_T& ioPartySize,
00102                        std::string& ioLogFilename) {
00103     // Default for the built-in input
00104     ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00105
00106     // Default for the inventory or schedule option
00107     ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00108
00109     // Declare a group of options that will be allowed only on command line
00110     boost::program_options::options_description generic ("Generic options");
00111     generic.add_options()
00112         ("prefix", "print installation prefix")
00113         ("version,v", "print version string")
00114         ("help,h", "produce help message");
00115
00116     // Declare a group of options that will be allowed both on command
00117     // line and in config file
00118
00119     boost::program_options::options_description config ("Configuration");
00120     config.add_options()
00121         ("builtin,b",
00122          "The sample BOM tree can be either built-in or parsed from an input file.
00123          That latter must then be given with the -i/--inventory or -s/--schedule option")
00124         ("for_schedule,f",
00125          "The BOM tree should be built from a schedule file (instead of from an
00126          inventory dump)")
00127         ("inventory,i",
00128          boost::program_options::value< std::string >(&ioInventoryFilename)->
00129          default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00130          "(CSV) input file for the inventory")
00131         ("schedule,s",
00132          boost::program_options::value< std::string >(&ioScheduleInputFilename)->
00133          default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00134          "(CSV) input file for the schedule")
00135         ("ond,o",
00136          boost::program_options::value< std::string >(&ioODInputFilename)->
00137          default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00138          "(CSV) input file for the O&D")
00139         ("yield,y",
00140          boost::program_options::value< std::string >(&ioYieldInputFilename)->
00141          default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00142          "(CSV) input file for the yield")
00143         ("segment_date_key,k",
00144          boost::program_options::value< std::string >(&ioSegmentDateKey)->
00145          default_value(K_AIRINV_DEFAULT_SEGMENT_DATE_KEY),
00146          "Segment-date key")
00147         ("class_code,c",
00148          boost::program_options::value< stdair::ClassCode_T >(&ioClassCode)->
00149          default_value(K_AIRINV_DEFAULT_CLASS_CODE),
00150          "Class code")
00151         ("party_size,p",
00152          boost::program_options::value< stdair::PartySize_T >(&ioPartySize)->
00153          default_value(K_AIRINV_DEFAULT_PARTY_SIZE),
00154          "Party size")
00155         ("log,l",
00156          boost::program_options::value< std::string >(&ioLogFilename)->
00157          default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00158          "Filename for the logs")
00159     ;
00160
00161     // Hidden options, will be allowed both on command line and

```

```

00152 // in config file, but will not be shown to the user.
00153 boost::program_options::options_description hidden ("Hidden options");
00154 hidden.add_options()
00155     ("copyright",
00156      boost::program_options::value< std::vector<std::string> >(),
00157       "Show the copyright (license)");
00158
00159 boost::program_options::options_description cmdline_options;
00160 cmdline_options.add(generic).add(config).add(hidden);
00161
00162 boost::program_options::options_description config_file_options;
00163 config_file_options.add(config).add(hidden);
00164 boost::program_options::options_description visible ("Allowed options");
00165 visible.add(generic).add(config);
00166
00167 boost::program_options::positional_options_description p;
00168 p.add ("copyright", -1);
00169
00170 boost::program_options::variables_map vm;
00171 boost::program_options::
00172     store (boost::program_options::command_line_parser (argc, argv).
00173            options (cmdline_options).positional(p).run(), vm);
00174
00175 std::ifstream ifs ("airinv.cfg");
00176 boost::program_options::store (parse_config_file (ifs, config_file_options),
00177                                vm);
00178 boost::program_options::notify (vm);
00179
00180 if (vm.count ("help")) {
00181     std::cout << visible << std::endl;
00182     return K_AIRINV_EARLY_RETURN_STATUS;
00183 }
00184
00185 if (vm.count ("version")) {
00186     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00187     return K_AIRINV_EARLY_RETURN_STATUS;
00188 }
00189
00190 if (vm.count ("prefix")) {
00191     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00192     return K_AIRINV_EARLY_RETURN_STATUS;
00193 }
00194
00195 if (vm.count ("builtin")) {
00196     ioIsBuiltin = true;
00197 }
00198 const std::string isBuiltinStr = (ioIsBuiltin == true)? "yes": "no";
00199 std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00200
00201 if (vm.count ("for_schedule")) {
00202     ioIsForSchedule = true;
00203 }
00204 const std::string isForScheduleStr = (ioIsForSchedule == true)? "yes": "no";
00205 std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00206           << std::endl;
00207
00208 if (ioIsBuiltin == false) {
00209
00210     if (ioIsForSchedule == false) {
00211         // The BOM tree should be built from parsing an inventory dump
00212         if (vm.count ("inventory")) {
00213             ioInventoryFilename = vm["inventory"].as< std::string >();
00214             std::cout << "Input inventory filename is: " << ioInventoryFilename
00215                     << std::endl;
00216         } else {
00217             // The built-in option is not selected. However, no inventory dump
00218             // file is specified
00219             std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00220                     << " -f/--for_schedule and -s/--schedule options "
00221                     << "must be specified" << std::endl;
00222         }
00223     } else {
00224         // The BOM tree should be built from parsing a schedule (and O&D) file
00225         if (vm.count ("schedule")) {
00226             ioScheduleInputFilename = vm["schedule"].as< std::string >();
00227             std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00228                     << std::endl;
00229         } else {
00230             // The built-in option is not selected. However, no schedule file
00231             // is specified
00232             std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00233                     << " -f/--for_schedule and -s/--schedule options "
00234                     << "must be specified" << std::endl;
00235         }
00236     }
00237 }
00238

```

```

00239
00240     if (vm.count ("ond")) {
00241         ioODInputFilename = vm["ond"].as< std::string >();
00242         std::cout << "Input O&D filename is: " << ioODInputFilename <<
std::endl;
00243     }
00244
00245     if (vm.count ("yield")) {
00246         ioYieldInputFilename = vm["yield"].as< std::string >();
00247         std::cout << "Input yield filename is: "
00248             << ioYieldInputFilename << std::endl;
00249     }
00250 }
00251 }
00252
00253 if (vm.count ("log")) {
00254     ioLogFilename = vm["log"].as< std::string >();
00255     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00256 }
00257
00258 return 0;
00259 }
00260
00261
00262 // ////////// M A I N //////////
00263 int main (int argc, char* argv[]) {
00264
00265     // State whether the BOM tree should be built-in or parsed from an
00266     // input file
00267     bool isBuiltin;
00268     bool isForSchedule;
00269
00270     // Input file names
00271     stdair::Filename_T lInventoryFilename;
00272     stdair::Filename_T lScheduleInputFilename;
00273     stdair::Filename_T lODInputFilename;
00274     stdair::Filename_T lYieldInputFilename;
00275
00276     // Parameters for the sale
00277     std::string lSegmentDateKey;
00278     stdair::ClassCode_T lClassCode;
00279     stdair::PartySize_T lPartySize;
00280
00281     // Output log File
00282     stdair::Filename_T lLogFilename;
00283
00284     // Call the command-line option parser
00285     const int lOptionParserStatus =
00286         readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
,
00287             lScheduleInputFilename, lODInputFilename,
00288             lYieldInputFilename, lSegmentDateKey, lClassCode,
00289             lPartySize, lLogFilename);
00290
00291     if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00292         return 0;
00293     }
00294
00295     // Set the log parameters
00296     std::ofstream logOutputFile;
00297     // Open and clean the log outputfile
00298     logOutputFile.open (lLogFilename.c_str());
00299     logOutputFile.clear();
00300
00301     // Initialise the inventory service
00302     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00303     AIRINV::AIRINV_Master_Service airinvService (lLogParams);
00304
00305     // DEBUG
00306     STDAIR_LOG_DEBUG ("Welcome to AirInv");
00307
00308     // Check whether or not a (CSV) input file should be read
00309     if (isBuiltin == true) {
00310
00311         // Build the sample BOM tree for RMOL
00312         airinvService.buildSampleBom();
00313
00314         // Define a specific segment-date key for the sample BOM tree
00315         //lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00316         lSegmentDateKey = "SQ,11,2010-02-08,SIN,BKK";
00317     } else {
00318         if (isForSchedule == true) {
00319             // Build the BOM tree from parsing a schedule file (and O&D list)
00320             AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00321             airinvService.parseAndLoad (lScheduleInputFilename, lODInputFilename,
00322                 lYieldFilePath);
00323

```



```

00324
00325     if (lSegmentDateKey == K_AIRINV_DEFAULT_SEGMENT_DATE_KEY) {
00326         // Define a specific segment-date key for the schedule-based inventory
00327         lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00328     }
00329
00330     } else {
00331         // Build the BOM tree from parsing an inventory dump file
00332         airinvService.parseAndLoad (lInventoryFilename);
00333     }
00334 }
00335
00336 // Make a booking
00337 const bool isSellSuccessful =
00338     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00339
00340 // DEBUG
00341 STDAIR_LOG_DEBUG ("Sale ('" << lSegmentDateKey << "', " << lClassCode << ": "
00342     << lPartySize << ") successful? " << isSellSuccessful);
00343
00344 // DEBUG: Display the whole BOM tree
00345 const std::string& lCSVDump = airinvService.csvDisplay();
00346 STDAIR_LOG_DEBUG (lCSVDump);
00347
00348 // Close the Log outputFile
00349 logOutputFile.close();
00350
00351 /*
00352     Note: as that program is not intended to be run on a server in
00353     production, it is better not to catch the exceptions. When it
00354     happens (that an exception is throwned), that way we get the
00355     call stack.
00356 */
00357
00358 return 0;
00359 }

```

25.31 airinv/bom/AirportList.hpp File Reference

```

#include <set>
#include <vector>
#include <stdair/stdair_basic_types.hpp>

```

Namespaces

- namespace [AIRINV](#)

Typedefs

- typedef std::set
< stdair::AirportCode_T > [AIRINV::AirportList_T](#)
- typedef std::vector
< stdair::AirportCode_T > [AIRINV::AirportOrderedList_T](#)

25.32 AirportList.hpp

```

00001 #ifndef __AIRINV_BOM_AIRPORTLIST_HPP
00002 #define __AIRINV_BOM_AIRPORTLIST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <set>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_basic_types.hpp>
00012
00013 namespace AIRINV {
00014
00016     typedef std::set<stdair::AirportCode_T> AirportList\_T;

```

```

00017  typedef std::vector<stdair::AirportCode_T> AirportOrderedList_T;
00018
00019  }
00020  #endif // __AIRINV_BOM_AIRPORTLIST_HPP

```

25.33 airinv/bom/BomAbstract.cpp File Reference

```
#include <airinv/bom/BomAbstract.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.34 BomAbstract.cpp

```

00001  // //////////////////////////////////////
00002  // Import section
00003  // //////////////////////////////////////
00004  // AIRINV
00005  #include <airinv/bom/BomAbstract.hpp>
00006
00007  namespace AIRINV {
00008
00009  }

```

25.35 airinv/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- class [AIRINV::BomAbstract](#)

Namespaces

- namespace [AIRINV](#)

Functions

- template<class charT , class traits >
std::basic_ostream< charT,
traits > & [operator<<](#) (std::basic_ostream< charT, traits > &ioOut, const [AIRINV::BomAbstract](#) &iBom)
- template<class charT , class traits >
std::basic_istream< charT,
traits > & [operator>>](#) (std::basic_istream< charT, traits > &ioIn, [AIRINV::BomAbstract](#) &ioBom)

25.35.1 Function Documentation

25.35.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & ioOut, const [AIRINV::BomAbstract](#) & iBom) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 56 of file [BomAbstract.hpp](#).

25.35.1.2 `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & ioln, AIRINV::BomAbstract & ioBom) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 84 of file [BomAbstract.hpp](#).

References [AIRINV::BomAbstract::fromStream\(\)](#).

25.36 BomAbstract.hpp

```

00001 #ifndef __AIRINV_BOM_BOMABSTRACT_HPP
00002 #define __AIRINV_BOM_BOMABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace AIRINV {
00012
00013     class BomAbstract {
00014     friend class FacBomAbstract;
00015     public:
00016         // ////////////////////////////////// Display support methods //////////////////////////////////
00017         virtual void toStream (std::ostream& ioOut) const = 0;
00018
00019         virtual void fromStream (std::istream& ioIn) = 0;
00020
00021         virtual std::string toString() const = 0;
00022
00023         virtual std::string describeKey() const = 0;
00024
00025         virtual std::string describeShortKey() const = 0;
00026
00027     protected:
00028         BomAbstract() {}
00029         BomAbstract(const BomAbstract&) {}
00030
00031         virtual ~BomAbstract() {}
00032     };
00033 }
00034
00035 template <class charT, class traits>
00036 inline
00037 std::basic_ostream<charT, traits>&
00038 operator<< (std::basic_ostream<charT, traits>& ioOut,
00039           const AIRINV::BomAbstract& iBom) {
00040     std::basic_ostringstream<charT, traits> ostr;
00041     ostr.copyfmt (ioOut);
00042     ostr.width (0);
00043
00044     // Fill string stream
00045     iBom.toStream (ostr);
00046
00047     // Print string stream
00048     ioOut << ostr.str();
00049
00050     return ioOut;
00051 }
00052
00053 template <class charT, class traits>
00054 inline
00055 std::basic_istream<charT, traits>&
00056 operator>> (std::basic_istream<charT, traits>& ioIn,
00057           AIRINV::BomAbstract& ioBom) {
00058     // Fill Bom object with input stream
00059     ioBom.fromStream (ioIn);
00060     return ioIn;
00061 }
00062
00063 #endif // __AIRINV_BOM_BOMABSTRACT_HPP

```

25.37 airinv/bom/BomRootHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <airinv/bom/BomRootHelper.hpp>
#include <airinv/bom/InventoryHelper.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.38 BomRootHelper.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BomManager.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/bom/Inventory.hpp>
00010 // AIRINV
00011 #include <airinv/bom/BomRootHelper.hpp>
00012 #include <airinv/bom/InventoryHelper.hpp>
00013
00014 namespace AIRINV {
00015 // //////////////////////////////////////
00016 void BomRootHelper::fillFromRouting (const stdair::BomRoot& iBomRoot) {
00017     const stdair::InventoryList_T& lInventoryList =
00018         stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00019
00020     // Browse the list of inventories and update each inventory.
00021     for (stdair::InventoryList_T::const_iterator itInventory =
00022         lInventoryList.begin();
00023         itInventory != lInventoryList.end(); ++itInventory) {
00024         const stdair::Inventory* lCurrentInventory_ptr = *itInventory;
00025         assert (lCurrentInventory_ptr != NULL);
00026         InventoryHelper::fillFromRouting (*lCurrentInventory_ptr);
00027     }
00028 }
00029
00030 }
```

25.39 airinv/bom/BomRootHelper.hpp File Reference

Classes

- class [AIRINV::BomRootHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.40 BomRootHelper.hpp

```
00001 #ifndef __AIRINV_BOM_BOMROOTHELPER_HPP
00002 #define __AIRINV_BOM_BOMROOTHELPER_HPP
00003
00004 // //////////////////////////////////////
```

```

00005 // Import section
00006 // //////////////////////////////////////
00007
00008 // Forward declarations.
00009 namespace stdair {
00010     class BomRoot;
00011 }
00012
00013 namespace AIRINV {
00014     class BomRootHelper {
00015     public:
00016         // ////////////////////////////////// Business Methods //////////////////////////////////
00017         static void fillFromRouting (const stdair::BomRoot&);
00018     };
00019 };
00020
00021 #endif // __AIRINV_BOM_BOMROOTHELPER_HPP

```

25.41 airinv/bom/BookingClassHelper.cpp File Reference

```

#include <cassert>
#include <stdair/bom/BookingClass.hpp>
#include <airinv/bom/BookingClassHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.42 BookingClassHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BookingClass.hpp>
00008 // AIRINV
00009 #include <airinv/bom/BookingClassHelper.hpp>
00010
00011 namespace AIRINV {
00012
00013 }

```

25.43 airinv/bom/BookingClassHelper.hpp File Reference

Classes

- class [AIRINV::BookingClassHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.44 BookingClassHelper.hpp

```

00001 #ifndef __AIRINV_BOM_BOOKINGCLASSHELPER_HPP
00002 #define __AIRINV_BOM_BOOKINGCLASSHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////

```

```

00007 // StdAir
00008 // #include <stdair/stdair_basic_types.hpp>
00009
00010 // Forward declarations
00011 namespace stdair {
00012     class BookingClass;
00013 }
00014
00015 namespace AIRINV {
00016
00019     class BookingClassHelper {
00020
00021     };
00022
00023 }
00024 #endif // __AIRINV_BOM_BOOKINGCLASSHELPER_HPP

```

25.45 airinv/bom/BookingClassStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <airinv/bom/BookingClassStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.46 BookingClassStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/BookingClass.hpp>
00010 // AirInv
00011 #include <airinv/bom/BookingClassStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 BookingClassStruct::BookingClassStruct () {
00017 }
00018
00019 // //////////////////////////////////////
00020 stdair::ClassCode_T BookingClassStruct::getFullSubclassCode() const {
00021     std::ostringstream ostr;
00022     ostr << _classCode << _subclassCode;
00023     return ostr.str();
00024 }
00025
00026 // //////////////////////////////////////
00027 const std::string BookingClassStruct::describe() const {
00028     std::ostringstream ostr;
00029     ostr << "          " << _classCode << _subclassCode
00030         << " (" << _parentClassCode << _parentSubclassCode << ")"
00031         << ", " << _cumulatedProtection << ":" << _protection
00032         << ", " << _nego
00033         << ", " << _noShowPercentage << ":" << _overbookingPercentage
00034         << ", " << _nbOfBookings << ":" << _nbOfGroupBookings
00035         << ":" << _nbOfPendingGroupBookings << ":" << _nbOfStaffBookings
00036         << ":" << _nbOfWLBookings << ":" << _etb
00037         << ", " << _netClassAvailability << ":" << _segmentAvailability
00038         << ":" << _netRevenueAvailability
00039         << std::endl;
00040     return ostr.str();
00041 }
00042
00043 // //////////////////////////////////////

```

```

00044 void BookingClassStruct::fill (stdair::BookingClass& ioBookingClass) const {
00045     // Set the Yield Range Upper Value
00046     // ioBookingClass.setYieldRangeValue (_yieldRangeUpperValue);
00047
00048     // Set the Availability
00049     // ioBookingClass.setAvailability (_availability);
00050
00051     // Set the number of seats
00052     // ioBookingClass.setNbOfSeats (_nbOfSeats);
00053
00054     // Set the Seat Index
00055     // ioBookingClass.setSeatIndex (_seatIndex);
00056 }
00057
00058 }

```

25.47 airinv/bom/BookingClassStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- struct [AIRINV::BookingClassStruct](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector
< BookingClassStruct > [AIRINV::BookingClassStructList_T](#)

25.48 BookingClassStruct.hpp

```

00001 #ifndef __AIRINV_BOM_BOOKINGCLASSSTRUCT_HPP
00002 #define __AIRINV_BOM_BOOKINGCLASSSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/AIRINV_Types.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class BookingClass;
00019 }
00020
00021 namespace AIRINV {
00022
00024     struct BookingClassStruct : public stdair::StructAbstract {
00025         // Attributes
00026         stdair::ClassCode_T _classCode;

```

```

00027     stdair::SubclassCode_T _subclassCode;
00028     stdair::ClassCode_T _parentClassCode;
00029     stdair::SubclassCode_T _parentSubclassCode;
00030     stdair::AuthorizationLevel_T _cumulatedProtection;
00031     stdair::AuthorizationLevel_T _protection;
00032     stdair::NbOfSeats_T _nego;
00033     stdair::OverbookingRate_T _noShowPercentage;
00034     stdair::OverbookingRate_T _overbookingPercentage;
00035     stdair::NbOfBookings_T _nbOfBookings;
00036     stdair::NbOfBookings_T _nbOfGroupBookings;
00037     stdair::NbOfBookings_T _nbOfPendingGroupBookings;
00038     stdair::NbOfBookings_T _nbOfStaffBookings;
00039     stdair::NbOfBookings_T _nbOfWLBookings;
00040     stdair::NbOfBookings_T _etb;
00041     stdair::Availability_T _netClassAvailability;
00042     stdair::Availability_T _segmentAvailability;
00043     stdair::Availability_T _netRevenueAvailability;
00044
00046     stdair::ClassCode_T getFullSubclassCode() const;
00047
00050     void fill (stdair::BookingClass&) const;
00051
00053     const std::string describe() const;
00054
00056     BookingClassStruct();
00057 };
00058
00060     typedef std::vector<BookingClassStruct> BookingClassStructList_T;
00061
00062 }
00063 #endif // __AIRINV_BOM_BUCKETSTRUCT_HPP

```

25.49 airinv/bom/BucketStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/Bucket.hpp>
#include <airinv/bom/BucketStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.50 BucketStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/Bucket.hpp>
00010 // AirInv
00011 #include <airinv/bom/BucketStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015     // //////////////////////////////////////
00016     BucketStruct::BucketStruct() {
00017     }
00018
00019     // //////////////////////////////////////
00020     const std::string BucketStruct::describe() const {
00021         std::ostringstream ostr;
00022         ostr << " " << _yieldRangeUpperValue << ":" << _availability
00023             << ":" << _nbOfSeats << ":" << _seatIndex
00024             << std::endl;
00025         return ostr.str();
00026     }
00027
00028     // //////////////////////////////////////

```



```

00029 void BucketStruct::fill (stdair::Bucket& ioBucket) const {
00030     // Set the Yield Range Upper Value
00031     ioBucket.setYieldRangeUpperValue (_yieldRangeUpperValue);
00032
00033     // Set the Availability
00034     ioBucket.setAvailability (_availability);
00035
00036     // Set the number of sold seats
00037     ioBucket.setSoldSeats (_nbOfSeats);
00038 }
00039
00040 }

```

25.51 airinv/bom/BucketStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- struct [AIRINV::BucketStruct](#)
Utility Structure for the parsing of Bucket structures.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector< BucketStruct > [AIRINV::BucketStructList_T](#)

25.52 BucketStruct.hpp

```

00001 #ifndef __AIRINV_BOM_BUCKETSTRUCT_HPP
00002 #define __AIRINV_BOM_BUCKETSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/AIRINV_Types.hpp>
00015
00017 namespace stdair {
00018     class Bucket;
00019 }
00020
00021 namespace AIRINV {
00022
00026     struct BucketStruct : public stdair::StructAbstract {
00027         // Attributes
00028         stdair::Yield_T _yieldRangeUpperValue;
00029         stdair::CabinCapacity_T _availability;
00030         stdair::NbOfSeats_T _nbOfSeats;
00031         stdair::SeatIndex_T _seatIndex;
00032

```

```

00034     void fill (stdair::Bucket&) const;
00035
00037     const std::string describe() const;
00038
00040     BucketStruct();
00041 };
00042
00044     typedef std::vector<BucketStruct> BucketStructList_T;
00045
00046 }
00047 #endif // __AIRINV_BOM_BUCKETSTRUCT_HPP

```

25.53 airinv/bom/DCPEventStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <vector>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/DCPEventStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.54 DCPEventStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <vector>
00008 // StdAir
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirInv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/bom/DCPEventStruct.hpp>
00014
00015 namespace AIRINV {
00016
00017 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018 DCPEventStruct::DCPEventStruct ()
00019 : _origin(""),
00020   _destination(""),
00021   _dateRangeStart(stdair::DEFAULT_DATE),
00022   _dateRangeEnd(stdair::DEFAULT_DATE),
00023   _timeRangeStart(stdair::DEFAULT_EPSILON_DURATION),
00024   _timeRangeEnd(stdair::DEFAULT_EPSILON_DURATION),
00025   _cabinCode(""),
00026   _pos(""),
00027   _advancePurchase(0),
00028   _saturdayStay("T"),
00029   _changeFees("T"),
00030   _nonRefundable("T"),
00031   _minimumStay(0),
00032   _DCP(0),
00033   _airlineCode(""),
00034   _classCode("") {
00035 }
00036
00037 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00038 stdair::Date_T DCPEventStruct::getDate() const {
00039     _itYear.check(); _itMonth.check(); _itDay.check();
00040     return stdair::Date_T (_itYear._value, _itMonth._value, _itDay._value);
00041 }
00042
00043 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00044 stdair::Duration_T DCPEventStruct::getTime() const {
00045     _itHours.check(); _itMinutes.check(); _itSeconds.check();

```

```

00046     return boost::posix_time::hours (_itHours._value)
00047         + boost::posix_time::minutes (_itMinutes._value)
00048         + boost::posix_time::seconds (_itSeconds._value);
00049 }
00050
00051
00052 // //////////////////////////////////////
00053 const std::string DCPEventStruct::describe () const {
00054     std::ostringstream ostr;
00055     ostr << "DCPEvent: "
00056         << _origin << "-" << _destination
00057         << ", POS(" << _pos << "), ["
00058         << _dateRangeStart << "/" << _dateRangeEnd << "]" - ["
00059         << boost::posix_time::to_simple_string(_timeRangeStart) << "/"
00060         << boost::posix_time::to_simple_string(_timeRangeEnd) << "]" \n      "
00061         << "-Cabin code- " << _cabinCode << "\n      "
00062         << "-Channel- " << _channel << "\n      "
00063         << "-Conditions- " << _saturdayStay << ", " << _changeFees << ", "
00064         << _nonRefundable << ", " << _advancePurchase << ", "
00065         << _minimumStay << "\n      "
00066         << "-DCP- " << _DCP << "\n      ";
00067     assert (_airlineCodeList.size() == _classCodeList.size());
00068     stdair::ClassList_StringList_T::const_iterator lItCurrentClassCode =
00069         _classCodeList.begin();
00070     stdair::AirlineCode_T lAirlineCode;
00071     std::string lClassCode;
00072     for (stdair::AirlineCodeList_T::const_iterator lItCurrentAirlineCode =
00073         _airlineCodeList.begin();
00074         lItCurrentAirlineCode != _airlineCodeList.end();
00075         lItCurrentAirlineCode++) {
00076         lAirlineCode = *lItCurrentAirlineCode;
00077         lClassCode = *lItCurrentClassCode;
00078         ostr << lAirlineCode << ", " << lClassCode;
00079         ostr << "
";
00080         lItCurrentClassCode++;
00081     }
00082     ostr << std::endl;
00083     return ostr.str();
00084 }
00085
00086 // //////////////////////////////////////
00087 const stdair::AirlineCode_T& DCPEventStruct::getFirstAirlineCode () const {
00088     assert (_airlineCodeList.size() > 0);
00089     stdair::AirlineCodeList_T::const_iterator itFirstAirlineCode =
00090         _airlineCodeList.begin();
00091     return *itFirstAirlineCode;
00092 }
00093
00094 // //////////////////////////////////////
00095 void DCPEventStruct::beginAirline () {
00096     _itCurrentAirlineCode = _airlineCodeList.begin();
00097 }
00098
00099 // //////////////////////////////////////
00100 bool DCPEventStruct::hasNotReachedEndAirline () const {
00101     bool result = (_itCurrentAirlineCode != _airlineCodeList.end());
00102     return result;
00103 }
00104
00105 // //////////////////////////////////////
00106 stdair::AirlineCode_T DCPEventStruct::getCurrentAirlineCode () const {
00107     assert (_itCurrentAirlineCode != _airlineCodeList.end());
00108     return (*_itCurrentAirlineCode);
00109 }
00110
00111 // //////////////////////////////////////
00112 void DCPEventStruct::iterateAirline () {
00113     if (_itCurrentAirlineCode != _classCodeList.end()) {
00114         _itCurrentAirlineCode++;
00115     }
00116 }
00117
00118 // //////////////////////////////////////
00119 const std::string& DCPEventStruct::getFirstClassCode () const {
00120     assert (_classCodeList.size() > 0);
00121     stdair::ClassList_StringList_T::const_iterator itFirstClassCode =
00122         _classCodeList.begin();
00123     return *itFirstClassCode;
00124 }
00125
00126 // //////////////////////////////////////
00127 void DCPEventStruct::beginClassCode () {
00128     _itCurrentClassCode = _classCodeList.begin();
00129 }
00130
00131 // //////////////////////////////////////
00132 bool DCPEventStruct::hasNotReachedEndClassCode () const {

```

```

00133     bool result = (_itCurrentClassCode != _classCodeList.end());
00134     return result;
00135 }
00136
00137 // //////////////////////////////////////
00138 std::string DCPEventStruct::getCurrentClassCode () const {
00139     assert (_itCurrentClassCode != _classCodeList.end());
00140     return (*_itCurrentClassCode);
00141 }
00142
00143 // //////////////////////////////////////
00144 void DCPEventStruct::iterateClassCode () {
00145     if (_itCurrentClassCode != _classCodeList.end()) {
00146         _itCurrentClassCode++;
00147     }
00148 }
00149 }
00150
00151 }
00152

```

25.55 airinv/bom/DCPEventStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/BasParserTypes.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- struct [AIRINV::DCPEventStruct](#)

Namespaces

- namespace [AIRINV](#)

25.56 DCPEventStruct.hpp

```

00001 #ifndef __AIRINV_BOM_DCPEVENTSTRUCT_HPP
00002 #define __AIRINV_BOM_DCPEVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_demand_types.hpp>
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/BasParserTypes.hpp>
00015 // AirInv
00016 #include <airinv/AIRINV_Types.hpp>
00017
00018 namespace AIRINV {
00019
00021     struct DCPEventStruct : public stdair::StructAbstract {
00022     public:
00023
00025         DCPEventStruct ();
00026
00028         stdair::Date_T getDate() const;
00029
00031         stdair::Duration_T getTime() const;
00032
00034         const std::string describe() const;

```

```

00035
00037     const unsigned int getAirlineListSize () const {
00038         return _airlineCodeList.size();
00039     }
00040
00042     const unsigned int getClassCodeListSize () const {
00043         return _classCodeList.size();
00044     }
00045
00047     const stdair::AirlineCode_T& getFirstAirlineCode () const;
00048
00052     void beginAirline ();
00053
00056     bool hasNotReachedEndAirline () const;
00057
00059     stdair::AirlineCode_T getCurrentAirlineCode () const;
00060
00063     void iterateAirline ();
00064
00066     const std::string& getFirstClassCode () const;
00067
00071     void beginClassCode ();
00072
00075     bool hasNotReachedEndClassCode () const;
00076
00078     std::string getCurrentClassCode () const;
00079
00082     void iterateClassCode ();
00083
00084 public:
00085     // ////////////////////////////////// Attributes //////////////////////////////////
00087     stdair::year_t _itYear;
00088     stdair::month_t _itMonth;
00089     stdair::day_t _itDay;
00090
00092     //long _itHours;
00093     stdair::hour_t _itHours;
00094     stdair::minute_t _itMinutes;
00095     stdair::second_t _itSeconds;
00096
00098     stdair::AirlineCodeList_T::iterator _itCurrentAirlineCode;
00099
00101     stdair::ClassList_StringList_T::iterator _itCurrentClassCode;
00102
00104     stdair::AirportCode_T _origin;
00105
00107     stdair::AirportCode_T _destination;
00108
00110     stdair::Date_T _dateRangeStart;
00111
00113     stdair::Date_T _dateRangeEnd;
00114
00116     stdair::Duration_T _timeRangeStart;
00117
00119     stdair::Duration_T _timeRangeEnd;
00120
00122     stdair::CabinCode_T _cabinCode;
00123
00125     stdair::CityCode_T _pos;
00126
00128     stdair::ChannelLabel_T _channel;
00129
00131     stdair::DayDuration_T _advancePurchase;
00132
00134     stdair::SaturdayStay_T _saturdayStay;
00135
00137     stdair::ChangeFees_T _changeFees;
00138
00140     stdair::NonRefundable_T _nonRefundable;
00141
00143     stdair::DayDuration_T _minimumStay;
00144
00146     stdair::PriceValue_T _DCP;
00147
00149     stdair::AirlineCode_T _airlineCode;
00150
00152     stdair::ClassCode_T _classCode;
00153
00155     stdair::AirlineCodeList_T _airlineCodeList;
00156
00158     //unsigned long int _nbOfAirlines;
00159
00161     stdair::ClassList_StringList_T _classCodeList;
00162
00163 };
00164
00165 }

```

```
00166 #endif // __AIRINV_BOM_DCPEVENTSTRUCT_HPP
```

25.57 airinv/bom/FareFamilyStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.58 FareFamilyStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/FareFamily.hpp>
00010 // AirInv
00011 #include <airinv/bom/FareFamilyStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 FareFamilyStruct::FareFamilyStruct()
00017 : _familyCode (stdair::DEFAULT_NULL_FARE_FAMILY_CODE),
00018   _classes (stdair::DEFAULT_NULL_CLASS_CODE) {
00019 }
00020
00021 // //////////////////////////////////////
00022 FareFamilyStruct::
00023 FareFamilyStruct (const stdair::FamilyCode_T& iFamilyCode,
00024                  const stdair::ClassList_String_T& iClasses)
00025 : _familyCode (iFamilyCode), _classes (iClasses) {
00026 }
00027
00028 // //////////////////////////////////////
00029 const std::string FareFamilyStruct::describe() const {
00030     std::ostringstream ostr;
00031
00032     ostr << "          " << _familyCode << " " << _classes << ", ";
00033
00034     for (BookingClassStructList_T::const_iterator itBkgClass= _classList.begin(
00035 );
00036         itBkgClass != _classList.end(); ++itBkgClass) {
00037         const BookingClassStruct& lBkgClass = *itBkgClass;
00038         ostr << lBkgClass.describe();
00039     }
00040     if (_classList.empty() == false) {
00041         ostr << std::endl;
00042     }
00043     return ostr.str();
00044 }
00045
00046 // //////////////////////////////////////
00047 void FareFamilyStruct::fill (stdair::FareFamily& ioFareFamily) const {
00048     // Set attributes
00049     // ioFareFamily.setSomeAttribute (_someAttribute);
00050 }
00051
00052 }
```

25.59 airinv/bom/FareFamilyStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/BookingClassStruct.hpp>
```

Classes

- struct [AIRINV::FareFamilyStruct](#)
Utility Structure for the parsing of fare family details.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector
< FareFamilyStruct > [AIRINV::FareFamilyStructList_T](#)

25.60 FareFamilyStruct.hpp

```
00001 #ifndef __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
00002 #define __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/BookingClassStruct.hpp>
00015
00016 namespace stdair {
00017     class FareFamily;
00018 }
00019
00020 namespace AIRINV {
00021
00022     struct FareFamilyStruct : public stdair::StructAbstract {
00023         // Attributes
00024         stdair::FamilyCode_T _familyCode;
00025         stdair::ClassList_String_T _classes;
00026         BookingClassStructList_T _classList;
00027
00028         FareFamilyStruct();
00029         FareFamilyStruct (const stdair::FamilyCode_T&,
00030                         const stdair::ClassList_String_T&);
00031
00032         void fill (stdair::FareFamily& const;
00033
00034         const std::string describe() const;
00035     };
00036
00037     typedef std::vector<FareFamilyStruct> FareFamilyStructList_T;
00038 }
00039 #endif // __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
```

25.61 airinv/bom/FlightDateHelper.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.62 FlightDateHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_Inventory.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/FlightDate.hpp>
00010 #include <stdair/bom/SegmentDate.hpp>
00011 #include <stdair/bom/SegmentCabin.hpp>
00012 #include <stdair/bom/LegCabin.hpp>
00013 // AIRINV
00014 #include <airinv/bom/FlightDateHelper.hpp>
00015 #include <airinv/bom/SegmentDateHelper.hpp>
00016 #include <airinv/bom/SegmentCabinHelper.hpp>
00017
00018 namespace AIRINV {
00019
00020 // //////////////////////////////////////
00021 void FlightDateHelper::
00022 updateBookingControls (stdair::FlightDate& ioFlightDate) {
00023
00024     // Parse the segment-cabin list and build the pseudo bid price vector.
00025     const stdair::SegmentDateList_T& LSDList =
00026         stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00027     for (stdair::SegmentDateList_T::const_iterator itSD = LSDList.begin();
00028          itSD != LSDList.end(); ++itSD) {
00029         const stdair::SegmentDate* LSD_ptr = *itSD;
00030         assert (LSD_ptr != NULL);
00031
00032         //
00033         const stdair::SegmentCabinList_T& LSCList =
00034             stdair::BomManager::getList<stdair::SegmentCabin> (*LSD_ptr);
00035         for (stdair::SegmentCabinList_T::const_iterator itSC = LSCList.begin();
00036              itSC != LSCList.end(); ++itSC) {
00037             stdair::SegmentCabin* LSC_ptr = *itSC;
00038             assert (LSC_ptr != NULL);
00039
00040             // Build the pseudo bid price vector for the segment-cabin.
00041             SegmentCabinHelper::buildPseudoBidPriceVector (*LSC_ptr);
00042
00043             // Update the booking controls using the pseudo bid price vector.
00044             SegmentCabinHelper::
00045                 updateBookingControlsUsingPseudoBidPriceVector (*LSC_ptr);
00046         }
00047     }
00048 }
00049
00050 // //////////////////////////////////////
00051 void FlightDateHelper::fillFromRouting(const stdair::FlightDate& iFlightDate)
00052 {
00053     const stdair::SegmentDateList_T& lSegmentDateList =
00054         stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00055 }

```



```

00055 // Browse the list of segment-dates and update each segment-date.
00056 for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00057     lSegmentDateList.begin();
00058     itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00059     stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00060     assert (lCurrentSegmentDate_ptr != NULL);
00061     SegmentDateHelper::fillFromRouting (*lCurrentSegmentDate_ptr);
00062 }
00063 }
00064
00065 // //////////////////////////////////////
00066 void FlightDateHelper::
00067 updateAvailabilityPool (const stdair::FlightDate& iFlightDate,
00068                        const stdair::CabinCode_T& iCabinCode){
00069     const stdair::SegmentDateList_T& lSegmentDateList =
00070         stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00071     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00072         lSegmentDateList.begin(); itSegmentDate != lSegmentDateList.end();
00073         ++itSegmentDate) {
00074         const stdair::SegmentDate* lSegmentDate_ptr = *itSegmentDate;
00075         assert (lSegmentDate_ptr != NULL);
00076         stdair::SegmentCabin& lSegmentCabin =
00077             stdair::BomManager::getObject<stdair::SegmentCabin> (*lSegmentDate_ptr,
00078                 iCabinCode);
00079
00080         // Update the availability pool of the segment-cabin to the minimal
00081         // availability pool of the member leg-cabins.
00082         const stdair::LegCabinList_T& lLegCabinList =
00083             stdair::BomManager::getList<stdair::LegCabin> (lSegmentCabin);
00084         stdair::Availability_T lAvailabilityPool = stdair::MAXIMAL_AVAILABILITY;
00085         for (stdair::LegCabinList_T::const_iterator itLegCabin =
00086             lLegCabinList.begin();
00087             itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00088             const stdair::LegCabin* lLegCabin_ptr = *itLegCabin;
00089             assert (lLegCabin_ptr != NULL);
00090             const stdair::Availability_T& lLegCabinAvailabilityPool =
00091                 lLegCabin_ptr->getAvailabilityPool();
00092             if (lAvailabilityPool > lLegCabinAvailabilityPool) {
00093                 lAvailabilityPool = lLegCabinAvailabilityPool;
00094             }
00095         }
00096         lSegmentCabin.setAvailabilityPool (lAvailabilityPool);
00097     }
00098 }
00099
00100 }

```

25.63 airinv/bom/FlightDateHelper.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class [AIRINV::FlightDateHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.64 FlightDateHelper.hpp

```

00001 #ifndef __AIRINV_BOM_FLIGHTDATEHELPER_HPP
00002 #define __AIRINV_BOM_FLIGHTDATEHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009

```

```

00010 // Forward declarations
00011 namespace stdair {
00012     class FlightDate;
00013 }
00014
00015 namespace AIRINV {
00016
00019     class FlightDateHelper {
00020     public:
00021         // ////////// Business Methods //////////
00024         static void fillFromRouting (const stdair::FlightDate&);
00025
00028         static void updateAvailabilityPool (const stdair::FlightDate&,
00029                                             const stdair::CabinCode_T&);
00030
00032         static void updateBookingControls (stdair::FlightDate&);
00033     };
00034
00035 }
00036 #endif // __AIRINV_BOM_FLIGHTDATEHELPER_HPP

```

25.65 airinv/bom/FlightDateStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/FlightDateStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.66 FlightDateStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AIRINV
00011 #include <airinv/AIRINV_Types.hpp>
00012 #include <airinv/bom/FlightDateStruct.hpp>
00013
00014 namespace AIRINV {
00015
00016     // //////////////////////////////////////
00017     FlightDateStruct::FlightDateStruct ()
00018     : _flightDate (stdair::DEFAULT_DATE),
00019       _flightTypeCode (FlightTypeCode::DOMESTIC),
00020       _flightVisibilityCode (FlightVisibilityCode::NORMAL),
00021       _itSeconds (0), _legAlreadyDefined (false) {
00022     }
00023
00024     // //////////////////////////////////////
00025     stdair::Date_T FlightDateStruct::getDate() const {
00026         return stdair::Date_T (_itYear + 2000, _itMonth, _itDay);
00027     }
00028
00029     // //////////////////////////////////////
00030     stdair::Duration_T FlightDateStruct::getTime() const {
00031         return boost::posix_time::hours (_itHours)
00032             + boost::posix_time::minutes (_itMinutes)
00033             + boost::posix_time::seconds (_itSeconds);
00034     }
00035
00036     // //////////////////////////////////////
00037     const std::string FlightDateStruct::describe() const {
00038         std::ostringstream ostr;

```

```

00039     ostr << _airlineCode << _flightNumber << ", " << _flightDate
00040     << " (" << _flightTypeCode;
00041     if (_flightVisibilityCode.getCode() != FlightVisibilityCode::NORMAL) {
00042         ostr << "/" << _flightVisibilityCode;
00043     }
00044     ostr << ")" << std::endl;
00045
00046     for (LegStructList_T::const_iterator itLeg = _legList.begin();
00047          itLeg != _legList.end(); ++itLeg) {
00048         const LegStruct& lLeg = *itLeg;
00049         ostr << lLeg.describe();
00050     }
00051
00052     for (SegmentStructList_T::const_iterator itSegment = _segmentList.begin();
00053          itSegment != _segmentList.end(); ++itSegment) {
00054         const SegmentStruct& lSegment = *itSegment;
00055         ostr << lSegment.describe();
00056     }
00057
00058     //ostr << "[Debug] - Staging Leg: ";
00059     //ostr << _itLeg.describe();
00060     //ostr << "[Debug] - Staging Cabin: ";
00061     //ostr << _itCabin.describe();
00062
00063     return ostr.str();
00064 }
00065
00066 // //////////////////////////////////////
00067 void FlightDateStruct::addAirport (const stdair::AirportCode_T& iAirport) {
00068     AirportList_T::const_iterator itAirport = _airportList.find (iAirport);
00069     if (itAirport == _airportList.end()) {
00070         // Add the airport code to the airport set
00071         const bool insertSuccessful = _airportList.insert (iAirport).second;
00072
00073         if (insertSuccessful == false) {
00074             // TODO: throw an exception
00075         }
00076
00077         // Add the airport code to the airport vector
00078         _airportOrderedList.push_back (iAirport);
00079     }
00080 }
00081
00082 // //////////////////////////////////////
00083 void FlightDateStruct::buildSegments () {
00084     // The list of airports encompasses all the airports on which
00085     // the flight takes off or lands. Moreover, that list is
00086     // time-ordered: the first airport is the initial departure of
00087     // the flight, and the last airport is the eventual point of
00088     // rest of the flight.
00089     // Be 1 the size of the ordered list of airports.
00090     // We want to generate all the segment combinations from the legs
00091     // and, hence, from all the possible (time-ordered) airport pairs.
00092     // Thus, we both iterator on i=0...l-1 and j=i+1...l
00093     assert (_airportOrderedList.size() >= 2);
00094
00095     _segmentList.clear();
00096     for (AirportOrderedList_T::const_iterator itAirport_i =
00097          _airportOrderedList.begin();
00098          itAirport_i != _airportOrderedList.end()-1; ++itAirport_i) {
00099         for (AirportOrderedList_T::const_iterator itAirport_j = itAirport_i + 1;
00100              itAirport_j != _airportOrderedList.end(); ++itAirport_j) {
00101             SegmentStruct lSegmentStruct;
00102             lSegmentStruct._boardingPoint = *itAirport_i;
00103             lSegmentStruct._offPoint = *itAirport_j;
00104             _segmentList.push_back (lSegmentStruct);
00105         }
00106     }
00107 }
00108
00109 // Clear the lists of airports, so that it is ready for the next flight
00110 _airportList.clear();
00111 _airportOrderedList.clear();
00112 }
00113
00114 // //////////////////////////////////////
00115 void FlightDateStruct::
00116 addSegmentCabin (const SegmentStruct& iSegment,
00117                  const SegmentCabinStruct& iCabin) {
00118     // Retrieve the Segment structure corresponding to the (boarding, off)
point
00119     // pair.
00120     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00121     for (; itSegment != _segmentList.end(); ++itSegment) {
00122         const SegmentStruct& lSegment = *itSegment;
00123
00124         const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;

```

```

00125     const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00126     if (lSegment._boardingPoint == lBoardingPoint
00127         && lSegment._offPoint == lOffPoint) {
00128         break;
00129     }
00130 }
00131
00132 // If the segment key (airport pair) given in the schedule input file
00133 // does not correspond to the leg (boarding, off) points, throw an
exception
00134 // so that the user knows the schedule input file is corrupted.
00135 if (itSegment == _segmentList.end()) {
00136     STDAIR_LOG_ERROR ("Within the inventory input file, there is a "
00137         << "flight for which the airports of segments "
00138         << "and those of the legs do not correspond.");
00139     throw SegmentDateNotFoundException ("Within the inventory input file, "
00140         "there is a flight for which the "
00141         "airports of segments and those of "
00142         "the legs do not correspond.");
00143 }
00144
00145 // Add the Cabin structure to the Segment Cabin structure.
00146 assert (itSegment != _segmentList.end());
00147 SegmentStruct& lSegment = *itSegment;
00148 lSegment._cabinList.push_back (iCabin);
00149 }
00150
00151 // //////////////////////////////////////
00152 void FlightDateStruct::
00153 addSegmentCabin (const SegmentCabinStruct& iCabin) {
00154     // Iterate on all the Segment structures (as they get the same cabin
00155     // definitions)
00156
00157     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00158         itSegment != _segmentList.end(); ++itSegment) {
00159         SegmentStruct& lSegment = *itSegment;
00160
00161         lSegment._cabinList.push_back (iCabin);
00162     }
00163 }
00164
00165 // //////////////////////////////////////
00166 void FlightDateStruct::
00167 addFareFamily (const SegmentStruct& iSegment,
00168     const SegmentCabinStruct& iCabin,
00169     const FareFamilyStruct& iFareFamily) {
00170     // Retrieve the Segment structure corresponding to the (boarding, off)
point
00171     // pair.
00172     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00173     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00174         const SegmentStruct& lSegment = *itSegment;
00175
00176         const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00177         const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00178         if (lSegment._boardingPoint == lBoardingPoint
00179             && lSegment._offPoint == lOffPoint) {
00180             break;
00181         }
00182     }
00183
00184 // If the segment key (airport pair) given in the schedule input file
00185 // does not correspond to the leg (boarding, off) points, throw an
exception
00186 // so that the user knows the schedule input file is corrupted.
00187 if (itSegment == _segmentList.end()) {
00188     STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00189         << "for which the airports of segments and "
00190         << "those of the legs do not correspond.");
00191     throw SegmentDateNotFoundException ("Within the schedule input file, "
00192         "there is a flight for which the "
00193         "airports of segments and those of "
00194         "the legs do not correspond.");
00195 }
00196
00197 // Add the Cabin structure to the Segment Cabin structure.
00198 assert (itSegment != _segmentList.end());
00199 SegmentStruct& lSegment = *itSegment;
00200
00201 // Retrieve the Segment cabin structure given the cabin code
00202 SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00203 for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00204     const SegmentCabinStruct& lCabin = *itCabin;
00205
00206     const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00207     if (iCabin._cabinCode == lCabinCode) {
00208         break;

```

```

00209     }
00210 }
00211
00212 // If the segmentCabin key (cabin code) given in the schedule input file
00213 // does not correspond to the stored cabin codes, throw an exception
00214 // so that the user knows the schedule input file is corrupted.
00215 if (itCabin == lSegment._cabinList.end()) {
00216     STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00217         << "for which the cabin code does not exist.");
00218     throw SegmentDateNotFoundException ("Within the schedule input file, "
00219         "there is a flight for which the "
00220         "cabin code does not exist.");
00221 }
00222
00223 // Add the Cabin structure to the Segment Cabin structure.
00224 assert (itCabin != lSegment._cabinList.end());
00225 SegmentCabinStruct& lCabin = *itCabin;
00226 lCabin._fareFamilies.push_back (iFareFamily);
00227 }
00228
00229 // //////////////////////////////////////
00230 void FlightDateStruct::
00231 addFareFamily (const SegmentCabinStruct& iCabin,
00232     const FareFamilyStruct& iFareFamily) {
00233     // Iterate on all the Segment structures (as they get the same cabin
00234     // definitions)
00235
00236     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00237         itSegment != _segmentList.end(); ++itSegment) {
00238         SegmentStruct& lSegment = *itSegment;
00239
00240         // Retrieve the Segment cabin structure given the cabin code
00241         SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00242         for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00243             const SegmentCabinStruct& lCabin = *itCabin;
00244
00245             const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00246             if (iCabin._cabinCode == lCabinCode) {
00247                 break;
00248             }
00249         }
00250
00251         // If the segmentCabin key (cabin code) given in the schedule input file
00252         // does not correspond to the stored cabin codes, throw an exception
00253         // so that the user knows the schedule input file is corrupted.
00254         if (itCabin == lSegment._cabinList.end()) {
00255             STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00256                 << "for which the cabin code does not exist.");
00257             throw SegmentDateNotFoundException ("Within the schedule input file, "
00258                 "there is a flight for which the "
00259                 "cabin code does not exist.");
00260         }
00261
00262         // Add the Cabin structure to the Segment Cabin structure.
00263         assert (itCabin != lSegment._cabinList.end());
00264         SegmentCabinStruct& lCabin = *itCabin;
00265         lCabin._fareFamilies.push_back (iFareFamily);
00266     }
00267 }
00268
00269 }

```

25.67 airinv/bom/FlightDateStruct.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airinv/basic/FlightTypeCode.hpp>
#include <airinv/basic/FlightVisibilityCode.hpp>
#include <airinv/bom/LegStruct.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
#include <airinv/bom/BucketStruct.hpp>
#include <airinv/bom/SegmentStruct.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
#include <airinv/bom/AirportList.hpp>

```

Classes

- struct [AIRINV::FlightDateStruct](#)

Namespaces

- namespace [AIRINV](#)

25.68 FlightDateStruct.hpp

```

00001 #ifndef __AIRINV_BOM_FLIGHTDATESTRUCT_HPP
00002 #define __AIRINV_BOM_FLIGHTDATESTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/StructAbstract.hpp>
00012 #include <stdair/bom/DoWStruct.hpp>
00013 // AirInv
00014 #include <airinv/basic/FlightTypeCode.hpp>
00015 #include <airinv/basic/FlightVisibilityCode.hpp>
00016 #include <airinv/bom/LegStruct.hpp>
00017 #include <airinv/bom/LegCabinStruct.hpp>
00018 #include <airinv/bom/BucketStruct.hpp>
00019 #include <airinv/bom/SegmentStruct.hpp>
00020 #include <airinv/bom/SegmentCabinStruct.hpp>
00021 #include <airinv/bom/FareFamilyStruct.hpp>
00022 #include <airinv/bom/AirportList.hpp>
00023
00024 namespace AIRINV {
00025
00027     struct FlightDateStruct : public stdair::StructAbstract {
00028
00030         stdair::Date_T getDate() const;
00031
00033         stdair::Duration_T getTime() const;
00034
00036         const std::string describe() const;
00037
00040         void addAirport (const stdair::AirportCode_T&);
00041
00043         void buildSegments ();
00044
00051         void addSegmentCabin (const SegmentStruct&,
00052                             const SegmentCabinStruct&);
00053
00059         void addSegmentCabin (const SegmentCabinStruct&);
00060
00067         void addFareFamily (const SegmentStruct&, const SegmentCabinStruct&,
00068                             const FareFamilyStruct&);
00069
00075         void addFareFamily (const SegmentCabinStruct&, const FareFamilyStruct&);
00076
00078         FlightDateStruct ();
00079
00080         // Attributes
00081         stdair::AirlineCode_T _airlineCode;
00082         stdair::FlightNumber_T _flightNumber;
00083         stdair::Date_T _flightDate;
00084         FlightTypeCode _flightTypeCode;
00085         FlightVisibilityCode _flightVisibilityCode;
00086         LegStructList_T _legList;
00087         SegmentStructList_T _segmentList;
00088
00090         unsigned int _itYear;
00091         unsigned int _itMonth;
00092         unsigned int _itDay;
00093         int _dateOffset;
00094
00096         long _itHours;
00097         long _itMinutes;
00098         long _itSeconds;

```

```

00099
00102     AirportList_T _airportList;
00103     AirportOrderedList_T _airportOrderedList;
00104
00107     bool _legAlreadyDefined;
00108     LegStruct _itLeg;
00109     LegCabinStruct _itLegCabin;
00110     BucketStruct _itBucket;
00111
00113     bool _areSegmentDefinitionsSpecific;
00114     SegmentStruct _itSegment;
00115     SegmentCabinStruct _itSegmentCabin;
00116     BookingClassStruct _itBookingClass;
00117 };
00118
00119 }
00120 #endif // __AIRINV_BOM_FLIGHTDATESTRUCT_HPP

```

25.69 airinv/bom/FlightPeriodStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.70 FlightPeriodStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AIRINV
00011 #include <airinv/AIRINV_Types.hpp>
00012 #include <airinv/bom/FlightPeriodStruct.hpp>
00013
00014 namespace AIRINV {
00015
00016 // //////////////////////////////////////
00017 FlightPeriodStruct::FlightPeriodStruct ()
00018 : _dateRange (stdair::BOOST_DEFAULT_DATE_PERIOD),
00019   _dow (stdair::DEFAULT_DOW_STRING),
00020   _legAlreadyDefined (false), _itSeconds (0) {
00021 }
00022
00023 // //////////////////////////////////////
00024 stdair::Date_T FlightPeriodStruct::getDate() const {
00025     return stdair::Date_T (_itYear, _itMonth, _itDay);
00026 }
00027
00028 // //////////////////////////////////////
00029 stdair::Duration_T FlightPeriodStruct::getTime() const {
00030     return boost::posix_time::hours (_itHours)
00031         + boost::posix_time::minutes (_itMinutes)
00032         + boost::posix_time::seconds (_itSeconds);
00033 }
00034
00035 // //////////////////////////////////////
00036 const std::string FlightPeriodStruct::describe() const {
00037     std::ostringstream ostr;
00038     ostr << _airlineCode << _flightNumber << ", " << _dateRange
00039         << " - " << _dow << std::endl;
00040
00041     for (LegStructList_T::const_iterator itLeg = _legList.begin();

```

```

00042         itLeg != _legList.end(); ++itLeg) {
00043             const LegStruct& lLeg = *itLeg;
00044             ostr << lLeg.describe();
00045         }
00046
00047         for (SegmentStructList_T::const_iterator itSegment = _segmentList.begin();
00048             itSegment != _segmentList.end(); ++itSegment) {
00049             const SegmentStruct& lSegment = *itSegment;
00050             ostr << lSegment.describe();
00051         }
00052
00053         //ostr << "[Debug] - Staging Leg: ";
00054         //ostr << _itLeg.describe();
00055         //ostr << "[Debug] - Staging Cabin: ";
00056         //ostr << _itCabin.describe();
00057
00058         return ostr.str();
00059     }
00060
00061     ///////////////////////////////////////////////////////////////////
00062 void FlightPeriodStruct::addAirport (const stdair::AirportCode_T& iAirport) {
00063     AirportList_T::const_iterator itAirport = _airportList.find (iAirport);
00064     if (itAirport == _airportList.end()) {
00065         // Add the airport code to the airport set
00066         const bool insertSuccessful = _airportList.insert (iAirport).second;
00067
00068         if (insertSuccessful == false) {
00069             // TODO: throw an exception
00070         }
00071
00072         // Add the airport code to the airport vector
00073         _airportOrderedList.push_back (iAirport);
00074     }
00075 }
00076
00077 ///////////////////////////////////////////////////////////////////
00078 void FlightPeriodStruct::buildSegments () {
00079     // The list of airports encompasses all the airports on which
00080     // the flight takes off or lands. Moreover, that list is
00081     // time-ordered: the first airport is the initial departure of
00082     // the flight, and the last airport is the eventual point of
00083     // rest of the flight.
00084     // Be l the size of the ordered list of airports.
00085     // We want to generate all the segment combinations from the legs
00086     // and, hence, from all the possible (time-ordered) airport pairs.
00087     // Thus, we both iterator on i=0...l-1 and j=i+1...l
00088     assert (_airportOrderedList.size() >= 2);
00089
00090     _segmentList.clear();
00091     for (AirportOrderedList_T::const_iterator itAirport_i =
00092         _airportOrderedList.begin();
00093         itAirport_i != _airportOrderedList.end()-1; ++itAirport_i) {
00094         for (AirportOrderedList_T::const_iterator itAirport_j = itAirport_i + 1;
00095             itAirport_j != _airportOrderedList.end(); ++itAirport_j) {
00096             SegmentStruct lSegmentStruct;
00097             lSegmentStruct._boardingPoint = *itAirport_i;
00098             lSegmentStruct._offPoint = *itAirport_j;
00099
00100             _segmentList.push_back (lSegmentStruct);
00101         }
00102     }
00103
00104     // Clear the lists of airports, so that it is ready for the next flight
00105     _airportList.clear();
00106     _airportOrderedList.clear();
00107 }
00108
00109 ///////////////////////////////////////////////////////////////////
00110 void FlightPeriodStruct::
00111 addSegmentCabin (const SegmentStruct& iSegment,
00112                 const SegmentCabinStruct& iCabin) {
00113     // Retrieve the Segment structure corresponding to the (boarding, off)
point
00114     // pair.
00115     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00116     for (; itSegment != _segmentList.end(); ++itSegment) {
00117         const SegmentStruct& lSegment = *itSegment;
00118
00119         const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00120         const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00121         if (lSegment._boardingPoint == lBoardingPoint
00122             && lSegment._offPoint == lOffPoint) {
00123             break;
00124         }
00125     }
00126
00127     // If the segment key (airport pair) given in the schedule input file

```



```

00128     // does not correspond to the leg (boarding, off) points, throw an
exception
00129     // so that the user knows the schedule input file is corrupted.
00130     if (itSegment == _segmentList.end()) {
00131         STDAIR_LOG_ERROR ("Within the schedule input file, there is a "
00132             << "flight for which the airports of segments "
00133             << "and those of the legs do not correspond.");
00134         throw SegmentDateNotFoundException ("Within the schedule input file, "
00135             "there is a flight for which the "
00136             "airports of segments and those of "
00137             "the legs do not correspond.");
00138     }
00139
00140     // Add the Cabin structure to the Segment Cabin structure.
00141     assert (itSegment != _segmentList.end());
00142     SegmentStruct& lSegment = *itSegment;
00143     lSegment._cabinList.push_back (iCabin);
00144 }
00145
00146 ///////////////////////////////////////////////////////////////////
00147 void FlightPeriodStruct::
00148 addSegmentCabin (const SegmentCabinStruct& iCabin) {
00149     // Iterate on all the Segment structures (as they get the same cabin
00150     // definitions)
00151     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00152         itSegment != _segmentList.end(); ++itSegment) {
00153         SegmentStruct& lSegment = *itSegment;
00154
00155         lSegment._cabinList.push_back (iCabin);
00156     }
00157 }
00158
00159 ///////////////////////////////////////////////////////////////////
00160 void FlightPeriodStruct::
00161 addFareFamily (const SegmentStruct& iSegment,
00162     const SegmentCabinStruct& iCabin,
00163     const FareFamilyStruct& iFareFamily) {
00164     // Retrieve the Segment structure corresponding to the (boarding, off)
point
00165     // pair.
00166     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00167     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00168         const SegmentStruct& lSegment = *itSegment;
00169
00170         const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00171         const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00172         if (lSegment._boardingPoint == lBoardingPoint
00173             && lSegment._offPoint == lOffPoint) {
00174             break;
00175         }
00176     }
00177
00178     // If the segment key (airport pair) given in the schedule input file
00179     // does not correspond to the leg (boarding, off) points, throw an
exception
00180     // so that the user knows the schedule input file is corrupted.
00181     if (itSegment == _segmentList.end()) {
00182         STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00183             << "for which the airports of segments and "
00184             << "those of the legs do not correspond.");
00185         throw SegmentDateNotFoundException ("Within the schedule input file, "
00186             "there is a flight for which the "
00187             "airports of segments and those of "
00188             "the legs do not correspond.");
00189     }
00190
00191     // Add the Cabin structure to the Segment Cabin structure.
00192     assert (itSegment != _segmentList.end());
00193     SegmentStruct& lSegment = *itSegment;
00194
00195     // Retrieve the Segment cabin structure given the cabin code
00196     SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00197     for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00198         const SegmentCabinStruct& lCabin = *itCabin;
00199
00200         const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00201         if (iCabin._cabinCode == lCabinCode) {
00202             break;
00203         }
00204     }
00205
00206     // If the segmentCabin key (cabin code) given in the schedule input file
00207     // does not correspond to the stored cabin codes, throw an exception
00208     // so that the user knows the schedule input file is corrupted.
00209     if (itCabin == lSegment._cabinList.end()) {
00210         STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00211             << "for which the cabin code does not exist.");

```

```

00212         throw SegmentDateNotFoundException ("Within the schedule input file, "
00213                                             "there is a flight for which the "
00214                                             "cabin code does not exist.");
00215     }
00216
00217     // Add the Cabin structure to the Segment Cabin structure.
00218     assert (itCabin != lSegment._cabinList.end());
00219     SegmentCabinStruct& lCabin = *itCabin;
00220     lCabin._fareFamilies.push_back(iFareFamily);
00221 }
00222
00223 // //////////////////////////////////////
00224 void FlightPeriodStruct::
00225 addFareFamily (const SegmentCabinStruct& iCabin,
00226               const FareFamilyStruct& iFareFamily) {
00227     // Iterate on all the Segment structures (as they get the same cabin
00228     // definitions)
00229
00230     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00231          itSegment != _segmentList.end(); ++itSegment) {
00232         SegmentStruct& lSegment = *itSegment;
00233
00234         // Retrieve the Segment cabin structure given the cabin code
00235         SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00236         for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00237             const SegmentCabinStruct& lCabin = *itCabin;
00238
00239             const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00240             if (iCabin._cabinCode == lCabinCode) {
00241                 break;
00242             }
00243         }
00244
00245         // If the segmentCabin key (cabin code) given in the schedule input file
00246         // does not correspond to the stored cabin codes, throw an exception
00247         // so that the user knows the schedule input file is corrupted.
00248         if (itCabin == lSegment._cabinList.end()) {
00249             STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight"
00250                             "<< " for which the cabin code does not exist.");
00251             throw SegmentDateNotFoundException ("Within the schedule input file, "
00252                                                 "there is a flight for which the "
00253                                                 "cabin code does not exist.");
00254         }
00255
00256         // Add the Cabin structure to the Segment Cabin structure.
00257         assert (itCabin != lSegment._cabinList.end());
00258         SegmentCabinStruct& lCabin = *itCabin;
00259         lCabin._fareFamilies.push_back(iFareFamily);
00260     }
00261 }
00262
00263 }

```

25.71 airinv/bom/FlightPeriodStruct.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
#include <airinv/bom/LegStruct.hpp>
#include <airinv/bom/SegmentStruct.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
#include <airinv/bom/AirportList.hpp>

```

Classes

- struct [AIRINV::FlightPeriodStruct](#)

Namespaces

- namespace [AIRINV](#)

25.72 FlightPeriodStruct.hpp

```

00001 #ifndef __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP
00002 #define __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/StructAbstract.hpp>
00012 #include <stdair/bom/DoWStruct.hpp>
00013 // AirInv
00014 #include <airinv/bom/LegCabinStruct.hpp>
00015 #include <airinv/bom/LegStruct.hpp>
00016 #include <airinv/bom/SegmentStruct.hpp>
00017 #include <airinv/bom/SegmentCabinStruct.hpp>
00018 #include <airinv/bom/FareFamilyStruct.hpp>
00019 #include <airinv/bom/AirportList.hpp>
00020
00021 namespace AIRINV {
00022
00024     struct FlightPeriodStruct : public stdair::StructAbstract {
00025
00027         stdair::Date_T getDate() const;
00028
00030         stdair::Duration_T getTime() const;
00031
00033         const std::string describe() const;
00034
00037         void addAirport (const stdair::AirportCode_T&);
00038
00040         void buildSegments ();
00041
00048         void addSegmentCabin (const SegmentStruct&,
00049                             const SegmentCabinStruct&);
00050
00056         void addSegmentCabin (const SegmentCabinStruct&);
00057
00064         void addFareFamily (const SegmentStruct&,
00065                             const SegmentCabinStruct&,
00066                             const FareFamilyStruct&);
00067
00073         void addFareFamily (const SegmentCabinStruct&,
00074                             const FareFamilyStruct&);
00075
00077         FlightPeriodStruct ();
00078
00079         // Attributes
00080         stdair::AirlineCode_T _airlineCode;
00081         stdair::FlightNumber_T _flightNumber;
00082         stdair::DatePeriod_T _dateRange;
00083         stdair::DoWStruct _dow;
00084         LegStructList_T _legList;
00085         SegmentStructList_T _segmentList;
00086
00089         bool _legAlreadyDefined;
00090         LegStruct _itLeg;
00091         LegCabinStruct _itLegCabin;
00092
00094         stdair::Date_T _dateRangeStart;
00095         stdair::Date_T _dateRangeEnd;
00096         unsigned int _itYear;
00097         unsigned int _itMonth;
00098         unsigned int _itDay;
00099         int _dateOffset;
00100
00102         long _itHours;
00103         long _itMinutes;
00104         long _itSeconds;
00105
00108         AirportList_T _airportList;
00109         AirportOrderedList_T _airportOrderedList;
00110
00112         bool _areSegmentDefinitionsSpecific;
00113         SegmentStruct _itSegment;
00114         SegmentCabinStruct _itSegmentCabin;

```

```

00115     };
00116
00117 }
00118 #endif // __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP

```

25.73 airinv/bom/GuillotineBlockHelper.cpp File Reference

```

#include <cassert>
#include <cmath>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/basic/BasConst_Curves.hpp>
#include <airinv/bom/GuillotineBlockHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.74 GuillotineBlockHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <cmath>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomRetriever.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014 #include <stdair/bom/BookingClass.hpp>
00015 #include <stdair/bom/GuillotineBlock.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 // AirInv
00018 #include <airinv/basic/BasConst_Curves.hpp>
00019 #include <airinv/bom/GuillotineBlockHelper.hpp>
00020 #include <airinv/bom/FlightDateHelper.hpp>
00021 #include <airinv/bom/SegmentCabinHelper.hpp>
00022
00023 namespace AIRINV {
00024
00025 // //////////////////////////////////////
00026 void GuillotineBlockHelper::
00027 takeSnapshots (stdair::GuillotineBlock& ioGuillotineBlock,
00028               const stdair::DateTime_T& iSnapshotTime) {
00029     // Retrieve the segment-cabin index and take the snapshots for
00030     // each segment-cabin.
00031     const stdair::SegmentCabinIndexMap_T& lSegmentCabinIndexMap =
00032         ioGuillotineBlock.getSegmentCabinIndexMap();
00033     for (stdair::SegmentCabinIndexMap_T::const_iterator itSCIdx =
00034         lSegmentCabinIndexMap.begin();
00035         itSCIdx != lSegmentCabinIndexMap.end(); ++itSCIdx) {
00036         const stdair::SegmentCabin* lSC_ptr = itSCIdx->first;
00037         assert (lSC_ptr != NULL);
00038         const stdair::BlockNumber_T& lSCIdx = itSCIdx->second;
00039
00040         const stdair::Date_T& lSnapshotDate = iSnapshotTime.date();

```

```

00041
00042     // Compare the date of the snapshot time and the departure date of
00043     // the segment-cabin in order to verify the necessity of taking
snapshots.
00044     const stdair::SegmentDate& lSegmentDate =
00045         stdair::BomManager::getParent<stdair::SegmentDate> (*lSC_ptr);
00046     const stdair::Date_T& lDepartureDate = lSegmentDate.getBoardingDate();
00047     const stdair::DateOffset_T lDateOffset = lDepartureDate - lSnapshotDate;
00048     const stdair::DTD_T lDTD = lDateOffset.days() + 1;
00049
00050     if (lDTD >= 0 && lDTD <= stdair::DEFAULT_MAX_DTD) {
00051         SegmentCabinHelper::updateAvailabilities (*lSC_ptr);
00052         takeSnapshots (ioGuillotineBlock, lDTD, *lSC_ptr, lSCIdx);
00053         registerProductAndPriceOrientedBookings (ioGuillotineBlock,
00054             lDTD, *lSC_ptr, lSCIdx);
00055     }
00056 }
00057 }
00058
00059 // //////////////////////////////////////
00060 void GuillotineBlockHelper::
00061 takeSnapshots (stdair::GuillotineBlock& ioGuillotineBlock,
00062     const stdair::DTD_T& iDTD,
00063     const stdair::SegmentCabin& iSegmentCabin,
00064     const stdair::BlockNumber_T iSegmentCabinIdx) {
00065
00066     // Extract the views for the corresponding DTD and segment-cabin.
00067     stdair::SegmentCabinDTDSnapshotView_T lBookingView = ioGuillotineBlock.
00068         getSegmentCabinDTDSnapshotView (iSegmentCabinIdx,
00069             iSegmentCabinIdx, iDTD);
00070     stdair::SegmentCabinDTDSnapshotView_T lCancellationView = ioGuillotineBlock
00071
00072         getSegmentCabinDTDCancellationSnapshotView (iSegmentCabinIdx,
00073             iSegmentCabinIdx, iDTD);
00074     stdair::SegmentCabinDTDSnapshotView_T lAvailabilityView = ioGuillotineBlock
00075
00076         getSegmentCabinDTDAvailabilitySnapshotView (iSegmentCabinIdx,
00077             iSegmentCabinIdx, iDTD);
00078
00079     // Retrieve the block index of the segment-cabin.
00080     std::ostream lSCMapKey;
00081     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00082         << iSegmentCabin.describeKey();
00083     const stdair::BlockIndex_T& lCabinIdx =
00084         ioGuillotineBlock.getBlockIndex (lSCMapKey.str());
00085     lAvailabilityView[lCabinIdx] = iSegmentCabin.getAvailabilityPool();
00086
00087     // Browse the booking class list
00088     const stdair::BookingClassList_T& lBCList =
00089         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00090     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00091         itBC != lBCList.end(); ++itBC) {
00092         const stdair::BookingClass* lBookingClass_ptr = *itBC;
00093         assert (lBookingClass_ptr != NULL);
00094
00095         // Retrieve the block index of the booking class.
00096         const stdair::BlockIndex_T& lIdx =
00097             ioGuillotineBlock.getBlockIndex (lBookingClass_ptr->describeKey());
00098
00099         // DEBUG
00100         // STDAIR_LOG_DEBUG ("Taking snapshot for "
00101             // << iSegmentCabin.describeKey() << ", "
00102             // << lBookingClass_ptr->describeKey()
00103             // << ", DTD: " << iDTD << ", nb of bookings: "
00104             // << lBookingClass_ptr->getNbOfBookings());
00105
00106         // Write the snapshot.
00107         lBookingView[lIdx]=lBookingClass_ptr->getNbOfBookings();
00108         lCancellationView[lIdx] =
00109             lBookingClass_ptr->getNbOfCancellations();
00110         lAvailabilityView[lIdx] =
00111             lBookingClass_ptr->getSegmentAvailability();
00112     }
00113 }
00114
00115 // //////////////////////////////////////
00116 void GuillotineBlockHelper::registerProductAndPriceOrientedBookings
00117 (stdair::GuillotineBlock& ioGuillotineBlock, const stdair::DTD_T& iDTD,
00118     const stdair::SegmentCabin& iSegmentCabin,
00119     const stdair::BlockNumber_T iSegmentCabinIdx) {
00120
00121     // Extract the views for the corresponding DTD and segment-cabin.
00122     stdair::SegmentCabinDTDRangeSnapshotView_T lRangeBookingView =
00123         ioGuillotineBlock.getSegmentCabinDTDRangeBookingSnapshotView (
00124             iSegmentCabinIdx, iSegmentCabinIdx, iDTD, iDTD + 1);
00125     stdair::SegmentCabinDTDRangeSnapshotView_T lRangeCancellationView =

```

```

00124         ioGuillotineBlock.getSegmentCabinDTDRangeCancellationSnapshotView (
00125             iSegmentCabinIdx, iSegmentCabinIdx, iDTD, iDTD + 1);
00126         stdair::iSegmentCabinDTDSnapshotView_T lProductAndPriceOrientedBookingView =
00127             ioGuillotineBlock.
00128             getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView (iSegmentCabinIdx, iSegmentCabinIdx, iDTD);
00129
00130         // Retrieve the block index of the segment-cabin.
00131         std::ostringstream lSCMapKey;
00132         lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00133             << iSegmentCabin.describeKey();
00134         const stdair::BlockIndex_T& lCabinIdx =
00135             ioGuillotineBlock.getBlockIndex (lSCMapKey.str());
00136
00137         // Retrieve the lowest class and treat the number of gross
00138         // bookings of this class the price oriented bookings.
00139         const stdair::BookingClassList_T& lBCList =
00140             stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00141         stdair::BookingClassList_T::const_reverse_iterator itBC = lBCList.rbegin();
00142         assert (itBC != lBCList.rend());
00143         stdair::BookingClass* lLowestClass_ptr = *itBC; ++itBC;
00144         assert (lLowestClass_ptr != NULL);
00145
00146         // Retrieve the block index of the booking class.
00147         const stdair::BlockIndex_T& lClassIdx =
00148             ioGuillotineBlock.getBlockIndex (lLowestClass_ptr->describeKey());
00149
00150         // Compute the number of gross bookings for this class.
00151         const stdair::NbOfBookings_T lNbOfNetBkgs =
00152             lRangeBookingView[lClassIdx][0] - lRangeBookingView[lClassIdx][1];
00153         const stdair::NbOfCancellations_T lNbOfCx =
00154             lRangeCancellationView[lClassIdx][0] - lRangeCancellationView[lClassIdx][1];
00155
00156         const stdair::NbOfBookings_T lNbOfGrossBkgs = lNbOfNetBkgs + lNbOfCx;
00157
00158         // Write this number of bookings to the price-oriented value.
00159         lProductAndPriceOrientedBookingView[lCabinIdx] = lNbOfGrossBkgs;
00160
00161         // Retrieve the lowest yield.
00162         const stdair::Yield_T& lLowestYield = lLowestClass_ptr->getYield();
00163
00164         // Boolean for "no lower class available" verification.
00165         bool noLowerClassAvl = true;
00166         if (lLowestClass_ptr->getSegmentAvailability() >= 1.0) {
00167             noLowerClassAvl = false;
00168         }
00169
00170         // Retrieve the FRAT5 coefficient and compute the sell-up coef.
00171         const double lFRAT5Coef = getFRAT5Coefficient (iDTD);
00172         const double lSellUpCoef = -log(0.5) / (lFRAT5Coef - 1);
00173
00174         // Browse the booking class list
00175         for (; itBC != lBCList.rend(); ++itBC) {
00176             const stdair::BookingClass* lBookingClass_ptr = *itBC;
00177             assert (lBookingClass_ptr != NULL);
00178
00179             // Retrieve the yield of the this class.
00180             const stdair::Yield_T& lYield = lBookingClass_ptr->getYield();
00181             assert (lYield > lLowestYield);
00182
00183             // Retrieve the block index of the booking class.
00184             const stdair::BlockIndex_T& lIdx =
00185                 ioGuillotineBlock.getBlockIndex (lBookingClass_ptr->describeKey());
00186
00187             // Compute the number of gross bookings for this class.
00188             const stdair::NbOfBookings_T lNetBkgs =
00189                 lRangeBookingView[lIdx][0] - lRangeBookingView[lIdx][1];
00190             const stdair::NbOfCancellations_T lCx =
00191                 lRangeCancellationView[lIdx][0] - lRangeCancellationView[lIdx][1];
00192             const stdair::NbOfBookings_T lGrossBkgs = lNetBkgs + lCx;
00193
00194             // If there is a lower class available, these gross bookings
00195             // will be considered product-oriented. Otherwise, they will be
00196             // considered price-oriented
00197             if (noLowerClassAvl == false) {
00198                 lProductAndPriceOrientedBookingView[lIdx] = lGrossBkgs;
00199             } else {
00200                 // Convert the bookings to Q-equivalent bookings.
00201                 const stdair::NbOfBookings_T lQEquiBkgs =
00202                     lGrossBkgs / exp ((1.0 - lYield/lLowestYield) * lSellUpCoef);
00203                 lProductAndPriceOrientedBookingView[lCabinIdx] += lQEquiBkgs;
00204
00205                 if (lBookingClass_ptr->getSegmentAvailability() >= 1.0) {
00206                     noLowerClassAvl = false;
00207                 }
00208             }
00209         }
00210     }
00211 }

```

```

00208
00209 // //////////////////////////////////////
00210 double GuillotineBlockHelper::getFRAT5Coefficient (const stdair::DTD_T& iDTD)
00211 {
00212     FRAT5Curve_T::const_iterator itFRAT5 =
00213         DEFAULT_PICKUP_FRAT5_CURVE.lower_bound (iDTD);
00214     assert (itFRAT5 != DEFAULT_PICKUP_FRAT5_CURVE.end());
00215     if (itFRAT5 == DEFAULT_PICKUP_FRAT5_CURVE.begin()) {
00216         return itFRAT5->second;
00217     }
00218     assert (itFRAT5 != DEFAULT_PICKUP_FRAT5_CURVE.begin());
00219     FRAT5Curve_T::const_iterator itNextFRAT5 = itFRAT5; --itNextFRAT5;
00220
00221     const stdair::DTD_T& lPrevDTD = itFRAT5->first;
00222     const stdair::DTD_T& lNextDTD = itNextFRAT5->first;
00223     const double& lPrevFRAT5 = itFRAT5->second;
00224     const double& lNextFRAT5 = itNextFRAT5->second;
00225     assert (lPrevDTD > lNextDTD);
00226
00227     double oFRAT5 = lPrevFRAT5
00228         + (iDTD - lNextDTD) * (lNextFRAT5 - lPrevFRAT5) / (lPrevDTD - lNextDTD);
00229
00230     return oFRAT5;
00231 }
00232 }
00233

```

25.75 airinv/bom/GuillotineBlockHelper.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>

```

Classes

- class [AIRINV::GuillotineBlockHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.76 GuillotineBlockHelper.hpp

```

00001 #ifndef __AIRINV_BOM_GUILLOTINEBLOCKHELPER_HPP
00002 #define __AIRINV_BOM_GUILLOTINEBLOCKHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     class GuillotineBlock;
00015     class SegmentCabin;
00016 }
00017
00018 namespace AIRINV {
00019
00022     class GuillotineBlockHelper {
00023     public:
00024         // ////////////////////////////////// Business Methods //////////////////////////////////
00026         static void takeSnapshots (stdair::GuillotineBlock&,
00027                                     const stdair::DateTime_T&);
00028     private:
00029         // ////////////////////////////////// Helpers for business methods. //////////////////////////////////

```

```

00031     static void takeSnapshots (stdair::GuillotineBlock&, const stdair::DTD_T&,
00032                               const stdair::SegmentCabin&,
00033                               const stdair::BlockNumber_T);
00034
00036     static void registerProductAndPriceOrientedBookings
00037     (stdair::GuillotineBlock&, const stdair::DTD_T&,
00038      const stdair::SegmentCabin&, const stdair::BlockNumber_T);
00039
00041     static double getFRAT5Coefficient (const stdair::DTD_T&);
00042 };
00043
00044 }
00045 #endif // __AIRINV_BOM_GUILLOTINEBLOCKHELPER_HPP

```

25.77 airinv/bom/InventoryHelper.cpp File Reference

```

#include <cassert>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/InventoryHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/GuillotineBlockHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.78 InventoryHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/bom/BomRetriever.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/Inventory.hpp>
00010 #include <stdair/bom/FlightDate.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014 #include <stdair/bom/BookingClass.hpp>
00015 #include <stdair/bom/GuillotineBlock.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 // AirInv
00020 #include <airinv/bom/InventoryHelper.hpp>
00021 #include <airinv/bom/FlightDateHelper.hpp>
00022 #include <airinv/bom/GuillotineBlockHelper.hpp>
00023 #include <airinv/bom/SegmentCabinHelper.hpp>
00024
00025 namespace AIRINV {
00026
00027 // //////////////////////////////////////
00028 void InventoryHelper::fillFromRouting (const stdair::Inventory& iInventory) {

```



```

00029     const stdair::FlightDateList_T& lFlightDateList =
00030         stdair::BomManager::getList<stdair::FlightDate> (iInventory);
00031
00032     // Browse the list of flight-dates and update each flight-date.
00033     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00034         lFlightDateList.begin();
00035         itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00036         const stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00037         assert (lCurrentFlightDate_ptr != NULL);
00038         FlightDateHelper::fillFromRouting (*lCurrentFlightDate_ptr);
00039     }
00040 }
00041
00042 // //////////////////////////////////////
00043 void InventoryHelper::
00044 calculateAvailability (const stdair::Inventory& iInventory,
00045                     const std::string& iFullSegmentDateKey,
00046                     stdair::TravelSolutionStruct& ioTravelSolution) {
00047
00048     // Create the map of class/availability for the given segment date.
00049     stdair::ClassAvailabilityMap_T lClassAvailabilityMap;
00050
00051     // DEBUG
00052     STDAIR_LOG_DEBUG (iFullSegmentDateKey);
00053     //
00054     stdair::SegmentDate* lSegmentDate_ptr =
00055         stdair::BomRetriever::retrieveSegmentDateFromLongKey(iInventory,
00056                                                             iFullSegmentDateKey)
00057 ;
00058     assert (lSegmentDate_ptr != NULL);
00059
00060     // Browse the segment-cabins and fill the map with the availability of
00061     // each booking class.
00062     const stdair::SegmentCabinList_T& lSegmentCabinList =
00063         stdair::BomManager::getList<stdair::SegmentCabin> (*lSegmentDate_ptr);
00064     for (stdair::SegmentCabinList_T::const_iterator itCabin =
00065         lSegmentCabinList.begin();
00066         itCabin != lSegmentCabinList.end(); ++itCabin) {
00067         stdair::SegmentCabin* lSegmentCabin_ptr = *itCabin;
00068         assert (lSegmentCabin_ptr != NULL);
00069
00070         // Compute the availability using the AU and the cumulative
00071         // booking counter.
00072         SegmentCabinHelper::updateAvailabilities (*lSegmentCabin_ptr);
00073         const stdair::BookingClassList_T& lBCList =
00074             stdair::BomManager::getList<stdair::BookingClass> (*lSegmentCabin_ptr);
00075         for (stdair::BookingClassList_T::const_reverse_iterator itBC =
00076             lBCList.rbegin(); itBC != lBCList.rend(); ++itBC) {
00077             stdair::BookingClass* lBC_ptr = *itBC;
00078             assert (lBC_ptr != NULL);
00079
00080             const stdair::Availability_T lAvl = lBC_ptr->getSegmentAvailability();
00081
00082             const stdair::ClassCode_T& lClassCode = lBC_ptr->getClassCode();
00083
00084             const bool insertSuccessful = lClassAvailabilityMap.
00085                 insert (stdair::ClassAvailabilityMap_T::value_type (lClassCode,
00086                                                                     lAvl)).second;
00086             assert (insertSuccessful == true);
00087         }
00088     }
00089
00090     //
00091     ioTravelSolution.addClassAvailabilityMap (lClassAvailabilityMap);
00092 }
00093
00094 // //////////////////////////////////////
00095 void InventoryHelper::
00096 getYieldAndBidPrice (const stdair::Inventory& iInventory,
00097                    const std::string& iFullSegmentDateKey,
00098                    stdair::TravelSolutionStruct& ioTravelSolution) {
00099
00100     // Create the map of class/availability for the given segment date.
00101     // stdair::ClassAvailabilityMap_T lClassAvailabilityMap;
00102
00103     stdair::ClassYieldMap_T lClassYieldMap;
00104
00105     stdair::ClassBpvMap_T lClassBpvMap;
00106
00107     // DEBUG
00108     STDAIR_LOG_DEBUG (iFullSegmentDateKey);
00109     //
00110     stdair::SegmentDate* lSegmentDate_ptr =
00111         stdair::BomRetriever::retrieveSegmentDateFromLongKey (iInventory,
00112                                                             iFullSegmentDateKey)

```

```

    );
00114     assert (lSegmentDate_ptr != NULL);
00115
00116     // Browse the segment-cabins and fill the maps with the bid price vector
reference
00117     // and yield of each booking class.
00118     const stdair::SegmentCabinList_T& lSegmentCabinList =
00119         stdair::BomManager::getList<stdair::SegmentCabin> (*lSegmentDate_ptr);
00120     for (stdair::SegmentCabinList_T::const_iterator itCabin =
00121         lSegmentCabinList.begin();
00122         itCabin != lSegmentCabinList.end(); ++itCabin) {
00123         stdair::SegmentCabin* lSegmentCabin_ptr = *itCabin;
00124         assert (lSegmentCabin_ptr != NULL);
00125
00126         stdair::BidPriceVector_T lBPV;
00127
00128
00129         //stdair::BidPriceVector_T lBPV;
00130         stdair::LegCabinList_T lLegCabinList =
00131             stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_ptr);
00132         assert (!lLegCabinList.empty());
00133         if (lLegCabinList.size() > 1) {
00134             // Compute the sum of bid prices and return a vector containing that
value.
00135             stdair::BidPrice_T lBidPriceValue = 0;
00136             for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begin(
);
00137                 itLC != lLegCabinList.end(); ++itLC) {
00138                 const stdair::LegCabin* lLegCabin_ptr = *itLC;
00139                 const stdair::BidPriceVector_T& lLegCabinBPV = lLegCabin_ptr->
getBidPriceVector();
00140                 if (!lLegCabinBPV.empty()) {
00141                     lBidPriceValue += lLegCabinBPV.back();
00142                 } else {
00143                     // If the remaining capacity is zero (empty bid price vector) on
one of the legs,
00144                     // then the remaining capacity of the segment is also zero (return
an empty bid price).
00145                     lBidPriceValue = std::numeric_limits<stdair::BidPrice_T>::max();
00146                     break;
00147                 }
00148             }
00149             if (lBidPriceValue < std::numeric_limits<stdair::BidPrice_T>::max()) {
00150                 lBPV.push_back(lBidPriceValue);
00151             }
00152
00153         } else {
00154             const stdair::LegCabin* lLegCabin_ptr = lLegCabinList.front();
00155             lBPV = lLegCabin_ptr->getBidPriceVector();
00156         }
00157
00158
00159         // const stdair::CabinCapacity_T& lCabinCapacity =
lSegmentCabin_ptr->getCapacity();
00160         // const stdair::CommittedSpace_T& lCommittedSpace =
lSegmentCabin_ptr->getCommittedSpace();
00161         // assert (lCabinCapacity - lCommittedSpace > 0);
00162         // lBPV.resize(lCabinCapacity - lCommittedSpace);
00163
00164         const stdair::Availability_T& lAvailabilityPool =
00165             lSegmentCabin_ptr->getAvailabilityPool();
00166         //assert (lAvailabilityPool > 0);
00167
00168         if (lAvailabilityPool < lBPV.size()) {
00169             lBPV.resize(lAvailabilityPool);
00170         }
00171
00172
00173         //
00174         ioTravelSolution.addBidPriceVector (lBPV);
00175
00176         const stdair::BidPriceVectorHolder_T& lBidPriceVectorHolder =
00177             ioTravelSolution.getBidPriceVectorHolder();
00178         const stdair::BidPriceVectorHolder_T::const_reverse_iterator itBPV =
00179             lBidPriceVectorHolder.rbegin();
00180         const stdair::BidPriceVector_T& lBpvRef = *itBPV;
00181
00182         const stdair::FareFamilyList_T& lFFList =
00183             stdair::BomManager::getList<stdair::FareFamily> (*lSegmentCabin_ptr);
00184         for (stdair::FareFamilyList_T::const_iterator itFF = lFFList.begin();
00185             itFF != lFFList.end(); ++itFF) {
00186             const stdair::FareFamily* lFareFamily_ptr = *itFF;
00187             assert (lFareFamily_ptr != NULL);
00188
00189             const stdair::BookingClassList_T& lBCList =
00190                 stdair::BomManager::getList<stdair::BookingClass> (*lFareFamily_ptr);
00191             for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();

```

```

00192         itBC != lBCList.end(); ++itBC) {
00193     const stdair::BookingClass* lBC_ptr = *itBC;
00194     assert (lBC_ptr != NULL);
00195
00196     const stdair::ClassCode_T& lClassCode = lBC_ptr->getClassCode();
00197
00198     const stdair::YieldValue_T lYld = lBC_ptr->getYld() ;
00199     const bool insertYieldMapSuccessful = lClassYieldMap.
00200         insert (stdair::ClassYieldMap_T::value_type (lClassCode,
00201             lYld)).second;
00202     assert (insertYieldMapSuccessful == true);
00203
00204     const bool insertBpvMapSuccessful = lClassBpvMap.
00205         insert (stdair::ClassBpvMap_T::value_type (lClassCode,
00206             &lBpvRef)).second;
00207     assert (insertBpvMapSuccessful == true);
00208
00209     // DEBUG
00210     // STDAIR_LOG_DEBUG ("Class: " << lClassCode
00211     //                  << ", " << "Yield: " << lYld << ", "
00212     //                  << "Bid price: " << lBpvRef.back() << ", "
00213     //                  << "Remaining capacity: "
00214     //                  << lCabinCapacity - lCommittedSpace);
00215
00216     //
00217     stdair::BidPrice_T lBpvVal = std::numeric_limits<double>::max();
00218     if (lBpvRef.empty() == false) {
00219         lBpvVal = lBpvRef.back();
00220     }
00221
00222     //lBpvVal = boost::lexical_cast<std::string> (lBpvRef.back());
00223     STDAIR_LOG_DEBUG ("Class: " << lClassCode
00224         << ", " << "Yield: " << lYld << ", "
00225         << "Bid price: " << lBpvVal << ", "
00226         << "Remaining capacity: " << lAvailabilityPool
00227         << " Segment date: " << iFullSegmentDateKey);
00228     }
00229 }
00230 }
00231
00232 //
00233 ioTravelSolution.addClassYieldMap (lClassYieldMap);
00234 ioTravelSolution.addClassBpvMap (lClassBpvMap);
00235 }
00236
00237
00238 ///////////////////////////////////////////////////////////////////
00239 bool InventoryHelper::sell (stdair::Inventory& ioInventory,
00240     const std::string& iFullSegmentDateKey,
00241     const stdair::ClassCode_T& iClassCode,
00242     const stdair::PartySize_T& iPartySize) {
00243     bool hasSaleBeenSuccessful = false;
00244
00245     // DEBUG
00246     STDAIR_LOG_DEBUG ("Full key: '" << iFullSegmentDateKey
00247         << "', " << iClassCode);
00248
00249     //
00250     stdair::BookingClass* lBookingClass_ptr =
00251         stdair::BomRetriever::retrieveBookingClassFromLongKey(ioInventory,
00252             iFullSegmentDateKey
00253
00254             iClassCode);
00255
00256     // DEBUG
00257     const std::string hasFoundBookingClassStr =
00258         (lBookingClass_ptr != NULL) ? "Yes" : "No";
00259     STDAIR_LOG_DEBUG ("Found booking class? " << hasFoundBookingClassStr);
00260
00261     if (lBookingClass_ptr != NULL) {
00262         // Register the sale in the class.
00263         lBookingClass_ptr->sell (iPartySize);
00264
00265         //
00266         stdair::FareFamily& lFareFamily =
00267             stdair::BomManager::getParent<stdair::FareFamily> (*lBookingClass_ptr);
00268
00269         //
00270         stdair::SegmentCabin& lSegmentCabin =
00271             stdair::BomManager::getParent<stdair::SegmentCabin> (lFareFamily);
00272
00273         //
00274         stdair::SegmentDate& lSegmentDate =
00275             stdair::BomManager::getParent<stdair::SegmentDate,
00276                 stdair::SegmentCabin> (lSegmentCabin);
00277
00278         //

```

```

00278     stdair::FlightDate& lFlightDate =
00279         stdair::BomManager::getParent<stdair::FlightDate,
00280             stdair::SegmentDate> (lSegmentDate);
00281
00282     // Update the committed space of the segment-cabins and the leg-cabins.
00283     SegmentCabinHelper::updateFromReservation (lFlightDate, lSegmentCabin,
00284         iPartySize);
00285
00286     // STDAIR_LOG_NOTIFICATION (lFlightDate.getDepartureDate()
00287     // << " " << iClassCode);
00288     hasSaleBeenSuccessful = true;
00289 }
00290
00291 return hasSaleBeenSuccessful;
00292 }
00293
00294 // //////////////////////////////////////
00295 bool InventoryHelper::cancel (stdair::Inventory& ioInventory,
00296     const std::string& iFullSegmentDateKey,
00297     const stdair::ClassCode_T& iClassCode,
00298     const stdair::PartySize_T& iPartySize) {
00299     bool hasCancellationBeenSuccessful = false;
00300
00301     // DEBUG
00302     STDAIR_LOG_DEBUG ("Full key: '" << iFullSegmentDateKey
00303         << "', " << iClassCode);
00304
00305     //
00306     stdair::BookingClass* lBookingClass_ptr =
00307         stdair::BomRetriever::retrieveBookingClassFromLongKey(ioInventory,
00308             iFullSegmentDateKey
00309             iClassCode);
00310
00311     // DEBUG
00312     const std::string hasFoundBookingClassStr =
00313         (lBookingClass_ptr != NULL) ? "Yes" : "No";
00314     STDAIR_LOG_DEBUG ("Found booking class? " << hasFoundBookingClassStr);
00315
00316     if (lBookingClass_ptr != NULL) {
00317         // Register the cancellation in the class.
00318         lBookingClass_ptr->cancel (iPartySize);
00319
00320         //
00321         stdair::FareFamily& lFareFamily =
00322             stdair::BomManager::getParent<stdair::FareFamily> (*lBookingClass_ptr);
00323
00324         //
00325         stdair::SegmentCabin& lSegmentCabin =
00326             stdair::BomManager::getParent<stdair::SegmentCabin> (lFareFamily);
00327
00328         //
00329         stdair::SegmentDate& lSegmentDate =
00330             stdair::BomManager::getParent<stdair::SegmentDate,
00331                 stdair::SegmentCabin> (lSegmentCabin);
00332
00333         //
00334         stdair::FlightDate& lFlightDate =
00335             stdair::BomManager::getParent<stdair::FlightDate,
00336                 stdair::SegmentDate> (lSegmentDate);
00337
00338         // Update the committed space of the segment-cabins and the leg-cabins.
00339         SegmentCabinHelper::updateFromReservation (lFlightDate, lSegmentCabin,
00340             -iPartySize);
00341
00342         // STDAIR_LOG_NOTIFICATION (lFlightDate.getDepartureDate()
00343         // << " " << iClassCode);
00344         hasCancellationBeenSuccessful = true;
00345     }
00346
00347     return hasCancellationBeenSuccessful;
00348 }
00349
00350 // //////////////////////////////////////
00351 void InventoryHelper::takeSnapshots(const stdair::Inventory& iInventory,
00352     const stdair::DateTime_T& iSnapshotTime)
00353 {
00354     // Browse the guillotine block list and take the snapshots for
00355     // each guillotine.
00356     const stdair::GuillotineBlockList_T& lGuillotineBlockList =
00357         stdair::BomManager::getList<stdair::GuillotineBlock> (iInventory);
00358     for (stdair::GuillotineBlockList_T::const_iterator itGB =
00359         lGuillotineBlockList.begin();
00360         itGB != lGuillotineBlockList.end(); ++itGB) {
00361         stdair::GuillotineBlock* lGuillotineBlock_ptr = *itGB;
00362         GuillotineBlockHelper::takeSnapshots(*lGuillotineBlock_ptr, iSnapshotTime)

```

```

    ;
00363     }
00364   }
00365 }
```

25.79 airinv/bom/InventoryHelper.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class [AIRINV::InventoryHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.80 InventoryHelper.hpp

```

00001 #ifndef __AIRINV_BOM_INVENTORYHELPER_HPP
00002 #define __AIRINV_BOM_INVENTORYHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     struct TravelSolutionStruct;
00015     class Inventory;
00016 }
00017
00018 namespace AIRINV {
00019
00022     class InventoryHelper {
00023     public:
00024         // ////////////////////////////////// Business Methods //////////////////////////////////
00027         static void fillFromRouting (const stdair::Inventory&);
00028
00030         static void calculateAvailability (const stdair::Inventory&,
00031                                         const std::string&,
00032                                         stdair::TravelSolutionStruct&);
00033
00035         static void getYieldAndBidPrice (const stdair::Inventory&,
00036                                         const std::string&,
00037                                         stdair::TravelSolutionStruct&);
00038
00040         static bool sell (stdair::Inventory&, const std::string& iSegmentDateKey,
00041                         const stdair::ClassCode_T&, const stdair::PartySize_T&);
00042
00044         static bool cancel (stdair::Inventory&, const std::string& iSegmentDateKey,
00045                           const stdair::ClassCode_T&, const stdair::PartySize_T&)
00046     ;
00048         static void takeSnapshots (const stdair::Inventory&,
00049                                   const stdair::DateTime_T&);
00050     };
00051
00052 }
00053 #endif // __AIRINV_BOM_INVENTORYHELPER_HPP
```

25.81 airinv/bom/LegCabinHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/LegCabinHelper.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.82 LegCabinHelper.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/LegCabin.hpp>
00008 // AIRINV
00009 #include <airinv/bom/LegCabinHelper.hpp>
00010
00011 namespace AIRINV {
00012
00013 }
```

25.83 airinv/bom/LegCabinHelper.hpp File Reference

Classes

- class [AIRINV::LegCabinHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.84 LegCabinHelper.hpp

```
00001 #ifndef __AIRINV_BOM_LEGCABINHELPER_HPP
00002 #define __AIRINV_BOM_LEGCABINHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 // Forward declarations
00009 namespace stdair {
00010     class LegCabin;
00011 }
00012
00013 namespace AIRINV {
00016     class LegCabinHelper {
00017     };
00018 };
00019
00020 }
00021 #endif // __AIRINV_BOM_LEGCABINHELPER_HPP
```

25.85 airinv/bom/LegCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.86 LegCabinStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/LegCabin.hpp>
00009 // AirInv
00010 #include <airinv/bom/LegCabinStruct.hpp>
00011
00012 namespace AIRINV {
00013
00014 // //////////////////////////////////////
00015 const std::string LegCabinStruct::describe() const {
00016     std::ostringstream ostr;
00017     ostr << " " << _cabinCode << ", " << _saleableCapacity
00018         << ", " << _adjustment << ", " << _dcsRegrade
00019         << ", " << _au << ", " << _avPool
00020         << ", " << _upr << ", " << _nbOfBookings << ", " << _nav
00021         << ", " << _gav << ", " << _acp << ", " << _etb
00022         << ", " << _staffNbOfBookings << ", " << _wlNbOfBookings
00023         << ", " << _groupNbOfBookings
00024         << std::endl;
00025
00026     for (BucketStructList_T::const_iterator itBucket = _bucketList.begin();
00027          itBucket != _bucketList.end(); ++itBucket) {
00028         const BucketStruct& lBucket = *itBucket;
00029         ostr << lBucket.describe();
00030     }
00031     if (_bucketList.empty() == false) {
00032         ostr << std::endl;
00033     }
00034     return ostr.str();
00035 }
00036
00037 // //////////////////////////////////////
00038 void LegCabinStruct::fill (stdair::LegCabin& ioLegCabin) const {
00039     // Set the Capacity
00040     ioLegCabin.setCapacities (_saleableCapacity);
00041 }
00042
00043 }
```

25.87 airinv/bom/LegCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/BucketStruct.hpp>
```

Classes

- struct [AIRINV::LegCabinStruct](#)

Namespaces

- namespace [stdair](#)
 Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector
 < LegCabinStruct > [AIRINV::LegCabinStructList_T](#)

25.88 LegCabinStruct.hpp

```

00001 #ifndef __AIRINV_BOM_LEGCABINSTRUCT_HPP
00002 #define __AIRINV_BOM_LEGCABINSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/BucketStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class LegCabin;
00019 }
00020
00021 namespace AIRINV {
00022
00024     struct LegCabinStruct : public stdair::StructAbstract {
00025         // Attributes
00026         stdair::CabinCode_T _cabinCode;
00027         stdair::CabinCapacity_T _saleableCapacity;
00028         stdair::CapacityAdjustment_T _adjustment;
00029         stdair::CapacityAdjustment_T _dcsRegrade;
00030         stdair::AuthorizationLevel_T _au;
00031         stdair::Availability_T _avPool;
00032         stdair::UPR_T _upr;
00033         stdair::NbOfBookings_T _nbOfBookings;
00034         stdair::Availability_T _nav;
00035         stdair::Availability_T _gav;
00036         stdair::OverbookingRate_T _acp;
00037         stdair::NbOfBookings_T _etb;
00038         stdair::NbOfBookings_T _staffNbOfBookings;
00039         stdair::NbOfBookings_T _wlNbOfBookings;
00040         stdair::NbOfBookings_T _groupNbOfBookings;
00041         BucketStructList_T _bucketList;
00042
00045         void fill (stdair::LegCabin&) const;
00046
00048         const std::string describe() const;
00049     };
00050
00052     typedef std::vector<LegCabinStruct> LegCabinStructList_T;
00053
00054 }
00055 #endif // __AIRINV_BOM_LEGCABINSTRUCT_HPP

```

25.89 airinv/bom/LegStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airinv/bom/LegStruct.hpp>

```


Namespaces

- namespace [AIRINV](#)

25.90 LegStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/LegDate.hpp>
00010 // AIRINV
00011 #include <airinv/bom/LegStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 LegStruct::LegStruct ()
00017 : _boardingDate (stdair::DEFAULT_DATE), _offDate (stdair::DEFAULT_DATE) {
00018 }
00019
00020 // //////////////////////////////////////
00021 const std::string LegStruct::describe() const {
00022     std::ostringstream ostr;
00023     ostr << " " << _boardingPoint << " / " << _boardingDate << " "
00024     << boost::posix_time::to_simple_string(_boardingTime)
00025     << " -- " << _offPoint << " / " << _offDate << " "
00026     << boost::posix_time::to_simple_string(_offTime)
00027     << " --> "
00028     << boost::posix_time::to_simple_string(_elapsed)
00029     << std::endl;
00030     for (LegCabinStructList_T::const_iterator itCabin = _cabinList.begin();
00031          itCabin != _cabinList.end(); itCabin++) {
00032         const LegCabinStruct& lCabin = *itCabin;
00033         ostr << lCabin.describe();
00034     }
00035     ostr << std::endl;
00036     return ostr.str();
00037 }
00038
00039 // //////////////////////////////////////
00040 void LegStruct::fill (const stdair::Date_T& iRefDate,
00041                     stdair::LegDate& ioLegDate) const {
00042     // Set the Off Point
00043     ioLegDate.setOffPoint (_offPoint);
00044     // Set the Boarding Date
00045     ioLegDate.setBoardingDate (iRefDate + _boardingDateOffset);
00046     // Set the Boarding Time
00047     ioLegDate.setBoardingTime (_boardingTime);
00048     // Set the Off Date
00049     ioLegDate.setOffDate (iRefDate + _offDateOffset);
00050     // Set the Off Time
00051     ioLegDate.setOffTime (_offTime);
00052     // Set the Elapsed Time
00053     ioLegDate.setElapsedTime (_elapsed);
00054 }
00055
00056 // //////////////////////////////////////
00057 void LegStruct::fill (stdair::LegDate& ioLegDate) const {
00058     // Set the Off Point
00059     ioLegDate.setOffPoint (_offPoint);
00060     // Set the Boarding Date
00061     ioLegDate.setBoardingDate (_offDate);
00062     // Set the Boarding Time
00063     ioLegDate.setBoardingTime (_boardingTime);
00064     // Set the Off Date
00065     ioLegDate.setOffDate (_offDate);
00066     // Set the Off Time
00067     ioLegDate.setOffTime (_offTime);
00068     // Set the Elapsed Time
00069     ioLegDate.setElapsedTime (_elapsed);
00070 }
00071 }
00072
00073 }

```

25.91 airinv/bom/LegStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
```

Classes

- struct [AIRINV::LegStruct](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector< LegStruct > [AIRINV::LegStructList_T](#)

25.92 LegStruct.hpp

```
00001 #ifndef __AIRINV_BOM_LEGSTRUCT_HPP
00002 #define __AIRINV_BOM_LEGSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AIRINV
00014 #include <airinv/bom/LegCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class LegDate;
00019 }
00020
00021 namespace AIRINV {
00022
00023     struct LegStruct : public stdair::StructAbstract {
00024         // Attributes
00025         stdair::AirportCode_T _boardingPoint;
00026         stdair::DateOffset_T _boardingDateOffset;
00027         stdair::Date_T _boardingDate;
00028         stdair::Duration_T _boardingTime;
00029         stdair::AirportCode_T _offPoint;
00030         stdair::DateOffset_T _offDateOffset;
00031         stdair::Date_T _offDate;
00032         stdair::Duration_T _offTime;
00033         stdair::Duration_T _elapsed;
00034         LegCabinStructList_T _cabinList;
00035
00036         void fill (const stdair::Date_T& iRefDate, stdair::LegDate&) const;
00042
00043         void fill (stdair::LegDate&) const;
00045
00046         const std::string describe() const;
00048
00049         LegStruct ();
00051     };
00052
00053     typedef std::vector<LegStruct> LegStructList_T;
00055
```

```

00056
00057 }
00058 #endif // __AIRINV_BOM_LEGSTRUCT_HPP

```

25.93 airinv/bom/SegmentCabinHelper.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.94 SegmentCabinHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/FlightDate.hpp>
00010 #include <stdair/bom/LegCabin.hpp>
00011 #include <stdair/bom/SegmentCabin.hpp>
00012 #include <stdair/bom/FareFamily.hpp>
00013 #include <stdair/bom/BookingClass.hpp>
00014 // AirInv
00015 #include <airinv/bom/SegmentCabinHelper.hpp>
00016 #include <airinv/bom/FlightDateHelper.hpp>
00017
00018 namespace AIRINV {
00019
00020 // //////////////////////////////////////
00021 void SegmentCabinHelper::initialiseAU (stdair::SegmentCabin& iSegmentCabin) {
00022
00023     // Initialise the capacity and availability pool.
00024     const stdair::LegCabinList_T& lLCList =
00025         stdair::BomManager::getList<stdair::LegCabin> (iSegmentCabin);
00026
00027     stdair::CabinCapacity_T lCapacity =
00028         std::numeric_limits<stdair::CabinCapacity_T>::max();
00029     for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00030          itLC != lLCList.end(); ++itLC) {
00031
00032         const stdair::LegCabin* lLC_ptr = *itLC;
00033         assert (lLC_ptr != NULL);
00034
00035         const stdair::CabinCapacity_T& lCabinCap = lLC_ptr->getOfferedCapacity();
00036         if (lCapacity > lCabinCap) {
00037             lCapacity = lCabinCap;
00038         }
00039     }
00040     iSegmentCabin.setCapacity (lCapacity);
00041     iSegmentCabin.setAvailabilityPool (lCapacity);
00042
00043     // Browse the list of booking classes and set the AU of each booking
00044     // class to the availability pool of the cabin.
00045     const stdair::BookingClassList_T& lBCList =
00046         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00047     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00048          itBC != lBCList.end(); ++itBC) {
00049         stdair::BookingClass* lBC_ptr = *itBC;

```

```

00050         assert (lBC_ptr != NULL);
00051         lBC_ptr->setAuthorizationLevel (lCapacity);
00052     }
00053 }
00054
00055 // //////////////////////////////////////
00056 void SegmentCabinHelper::
00057 updateFromReservation (const stdair::FlightDate& iFlightDate,
00058                       stdair::SegmentCabin& ioSegmentCabin,
00059                       const stdair::PartySize_T& iNbOfBookings){
00060     // Update the committed space of the segment-cabin.
00061     ioSegmentCabin.updateFromReservation (iNbOfBookings);
00062
00063     // Update the committed space of the member leg-cabins.
00064     const stdair::LegCabinList_T& lLegCabinList =
00065         stdair::BomManager::getList<stdair::LegCabin> (ioSegmentCabin);
00066     for (stdair::LegCabinList_T::const_iterator itLegCabin =
00067         lLegCabinList.begin();
00068         itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00069         stdair::LegCabin* lLegCabin_ptr = *itLegCabin;
00070         assert (lLegCabin_ptr != NULL);
00071         lLegCabin_ptr->updateFromReservation (iNbOfBookings);
00072     }
00073
00074     // Update the availability pool of all the segment-cabin which belong to
the
00075     // same flight-date.
00076     const stdair::CabinCode_T& lCabinCode = ioSegmentCabin.getCabinCode();
00077     FlightDateHelper::updateAvailabilityPool (iFlightDate, lCabinCode);
00078 }
00079
00080 // //////////////////////////////////////
00081 void SegmentCabinHelper::
00082 buildPseudoBidPriceVector (stdair::SegmentCabin& ioSegmentCabin) {
00083     // Retrieve the segment-cabin capacity.
00084     const stdair::Availability_T& lAvlPool=ioSegmentCabin.getAvailabilityPool()
;
00085     const unsigned int lAvlPoolInt =
00086         static_cast<unsigned int> (lAvlPool);
00087     stdair::BidPriceVector_T lPseudoBidPriceVector (lAvlPoolInt, 0.0);
00088
00089     // Browse the leg-cabin list.
00090     const stdair::LegCabinList_T& lLCList =
00091         stdair::BomManager::getList<stdair::LegCabin> (ioSegmentCabin);
00092     for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00093         itLC != lLCList.end(); ++itLC) {
00094         const stdair::LegCabin* lLC_ptr = *itLC;
00095         assert (lLC_ptr != NULL);
00096
00097         const stdair::BidPriceVector_T& lBPV = lLC_ptr->getBidPriceVector();
00098         stdair::BidPriceVector_T::const_reverse_iterator itBP = lBPV.rbegin();
00099         for (stdair::BidPriceVector_T::reverse_iterator itPBP =
00100             lPseudoBidPriceVector.rbegin();
00101             itPBP != lPseudoBidPriceVector.rend(); ++itPBP, ++itBP) {
00102             assert (itBP != lBPV.rend());
00103             stdair::BidPrice_T& lCurrentPBP = *itPBP;
00104             const stdair::BidPrice_T& lCurrentBP = *itBP;
00105             lCurrentPBP += lCurrentBP;
00106         }
00107     }
00108
00109     ioSegmentCabin.setBidPriceVector (lPseudoBidPriceVector);
00110
00111     // // DEBUG
00112     // std::ostream ostr;
00113     // ostr << "Pseudo BPV: ";
00114     // for (stdair::BidPriceVector_T::const_iterator itBP =
00115         // lPseudoBidPriceVector.begin(); itBP !=
lPseudoBidPriceVector.end();
00116         // ++itBP) {
00117         //     const stdair::BidPrice_T& lCurrentBP = *itBP;
00118         //     ostr << lCurrentBP << " ";
00119         // }
00120     // // STDAIR_LOG_DEBUG (ostr.str());
00121     // std::cout << ostr.str() << std::endl;
00122 }
00123
00124 // //////////////////////////////////////
00125 void SegmentCabinHelper::
00126 updateBookingControlsUsingPseudoBidPriceVector (const stdair::SegmentCabin&
ioSegmentCabin) {
00127     // Retrieve the pseudo bid price vector.
00128     const stdair::BidPriceVector_T& lPseudoBPV =
00129         ioSegmentCabin.getBidPriceVector();
00130     const stdair::Availability_T& lAvlPool=ioSegmentCabin.getAvailabilityPool();
00131
00132     // Update the cumulative booking limit for all booking classes.

```

```

00133     const stdair::BookingClassList_T& lBCList =
00134         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00135     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00136          itBC != lBCList.end(); ++itBC) {
00137         stdair::BookingClass* lBC_ptr = *itBC;
00138         assert (lBC_ptr != NULL);
00139
00140         lBC_ptr->setCumulatedBookingLimit (lAvlPool);
00141         const stdair::Yield_T& lYield = lBC_ptr->getYield();
00142         for (stdair::BidPriceVector_T::const_reverse_iterator itBP =
00143              lPseudoBPV.rbegin(); itBP != lPseudoBPV.rend(); ++itBP) {
00144             const stdair::BidPrice_T& lBP = *itBP;
00145             if (lYield < lBP) {
00146                 stdair::BookingLimit_T lCumBL = itBP - lPseudoBPV.rbegin();
00147                 lBC_ptr->setCumulatedBookingLimit (lCumBL);
00148                 break;
00149             }
00150         }
00151     }
00152
00153     // Update the authorization levels from the booking limits
00154     updateAUS (iSegmentCabin);
00155 }
00156
00157 // //////////////////////////////////////
00158 void SegmentCabinHelper::updateAUS(const stdair::SegmentCabin& iSegmentCabin)
00159 {
00160     // Browse the booking class list and compute the AU from the
00161     // cumulative booking counter and the cumulative booking limit.
00162     stdair::NbOfBookings_T lCumulativeBookingCounter = 0.0;
00163     const stdair::BookingClassList_T& lBCList =
00164         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00165     for (stdair::BookingClassList_T::const_reverse_iterator itBC =
00166          lBCList.rbegin(); itBC != lBCList.rend(); ++itBC) {
00167         stdair::BookingClass* lBC_ptr = *itBC;
00168         assert (lBC_ptr != NULL);
00169
00170         const stdair::NbOfBookings_T& lNbOfBookings = lBC_ptr->getNbOfBookings();
00171         lCumulativeBookingCounter += lNbOfBookings;
00172
00173         const stdair::BookingLimit_T& lCumBookingLimit =
00174             lBC_ptr->getCumulatedBookingLimit();
00175
00176         stdair::AuthorizationLevel_T lAU =
00177             lCumulativeBookingCounter + lCumBookingLimit;
00178         lBC_ptr->setAuthorizationLevel (lAU);
00179
00180         // DEBUG
00181         // STDAIR_LOG_DEBUG ("Updating the AU for class: "
00182                             // << lBC_ptr->describeKey()
00183                             // << ", with BL: " << lCumBookingLimit
00184                             // << ", CumuBkg: " << lCumulativeBookingCounter
00185                             // << ", AU: " << lAU);
00186     }
00187 }
00188
00189 // //////////////////////////////////////
00190 void SegmentCabinHelper::
00191 updateAvailabilities (const stdair::SegmentCabin& iSegmentCabin) {
00192     // Browse the booking class list and compute the avl from the
00193     // cumulative booking counter and the AU.
00194     stdair::NbOfBookings_T lCumulativeBookingCounter = 0.0;
00195     const stdair::BookingClassList_T& lBCList =
00196         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00197     for (stdair::BookingClassList_T::const_reverse_iterator itBC =
00198          lBCList.rbegin(); itBC != lBCList.rend(); ++itBC) {
00199         stdair::BookingClass* lBC_ptr = *itBC;
00200         assert (lBC_ptr != NULL);
00201
00202         const stdair::NbOfBookings_T& lNbOfBookings = lBC_ptr->getNbOfBookings();
00203         lCumulativeBookingCounter += lNbOfBookings;
00204
00205         const stdair::AuthorizationLevel_T& lAU=lBC_ptr->getAuthorizationLevel();
00206
00207         const stdair::Availability_T lAvl = lAU - lCumulativeBookingCounter;
00208         lBC_ptr->setSegmentAvailability (lAvl);
00209     }
00210 }

```

25.95 airinv/bom/SegmentCabinHelper.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class [AIRINV::SegmentCabinHelper](#)
Class representing the actual business functions for an airline segment-cabin.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.96 SegmentCabinHelper.hpp

```

00001 #ifndef __AIRINV_BOM_SEGMENTCABINHELPER_HPP
00002 #define __AIRINV_BOM_SEGMENTCABINHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009
00010 // Forward declarations
00011 namespace stdair {
00012     class FlightDate;
00013     class SegmentCabin;
00014     class FareFamily;
00015 }
00016
00017 namespace AIRINV {
00018
00023     class SegmentCabinHelper {
00024     public:
00025         // ////////////////////////////////// Business Methods //////////////////////////////////
00029         static void updateFromReservation (const stdair::FlightDate&,
00030                                           stdair::SegmentCabin&,
00031                                           const stdair::PartySize_T&);
00032
00036         static void buildPseudoBidPriceVector (stdair::SegmentCabin&);
00037
00041         static void updateBookingControlsUsingPseudoBidPriceVector (const
stdair::SegmentCabin&);
00042
00045         static void updateAUs (const stdair::SegmentCabin&);
00046
00049         static void updateAvailabilities (const stdair::SegmentCabin&);
00050
00054         static void initialiseAU (stdair::SegmentCabin&);
00055     };
00056
00057 }
00058 #endif // __AIRINV_BOM_SEGMENTCABINHELPER_HPP

```

25.97 airinv/bom/SegmentCabinStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentCabin.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.98 SegmentCabinStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/SegmentCabin.hpp>
00009 // AirInv
00010 #include <airinv/bom/SegmentCabinStruct.hpp>
00011
00012 namespace AIRINV {
00013
00014 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00015 const std::string SegmentCabinStruct::describe() const {
00016     std::ostringstream ostr;
00017
00018     ostr << "          " << _cabinCode << ", " << _nbOfBookings << std::endl;
00019
00020     for (FareFamilyStructList_T::const_iterator itFF = _fareFamilies.begin();
00021          itFF != _fareFamilies.end(); ++itFF) {
00022         const FareFamilyStruct& lFF = *itFF;
00023         ostr << lFF.describe();
00024     }
00025     if (_fareFamilies.empty() == false) {
00026         ostr << std::endl;
00027     }
00028
00029     return ostr.str();
00030 }
00031
00032 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00033 void SegmentCabinStruct::fill (stdair::SegmentCabin& ioSegmentCabin) const {
00034     // Set the total number of bookings
00035     // ioSegmentCabin.setNbOfBookings (_nbOfBookings);
00036 }
00037
00038 }

```

25.99 airinv/bom/SegmentCabinStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>

```

Classes

- struct [AIRINV::SegmentCabinStruct](#)
Utility Structure for the parsing of SegmentCabin details.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::vector
 < SegmentCabinStruct > [AIRINV::SegmentCabinStructList_T](#)

25.100 SegmentCabinStruct.hpp

```

00001 #ifndef __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP
00002 #define __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/FareFamilyStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentCabin;
00019 }
00020
00021 namespace AIRINV {
00022
00026     struct SegmentCabinStruct : public stdair::StructAbstract {
00027         // Attributes
00028         stdair::CabinCode_T _cabinCode;
00029         stdair::NbOfBookings_T _nbOfBookings;
00030         FareFamilyStruct _itFareFamily;
00031         FareFamilyStructList_T _fareFamilies;
00032
00037         void fill (stdair::SegmentCabin& const;
00038
00042         const std::string describe() const;
00043     };
00044
00048     typedef std::vector<SegmentCabinStruct> SegmentCabinStructList_T;
00049
00050 }
00051 #endif // __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP

```

25.101 airinv/bom/SegmentDateHelper.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airinv/bom/SegmentDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.102 SegmentDateHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_General.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/SegmentDate.hpp>
00010 #include <stdair/bom/SegmentCabin.hpp>
00011 #include <stdair/bom/LegDate.hpp>
00012 // AIRINV
00013 #include <airinv/bom/SegmentDateHelper.hpp>
00014 #include <airinv/bom/SegmentCabinHelper.hpp>

```



```

00015
00016 namespace AIRINV {
00017 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018 void SegmentDateHelper::fillFromRouting (stdair::SegmentDate& ioSegmentDate)
00019 {
00020     /*
00021      * If the segment is just marketed by this carrier,
00022      * retrieve the operating segment and call the fillFromRouting
00023      * method on it.
00024      */
00025     stdair::SegmentDate* lOperatingSegmentDate_ptr =
00026         ioSegmentDate.getOperatingSegmentDate ();
00027     if (lOperatingSegmentDate_ptr != NULL) {
00028         fillFromRouting (*lOperatingSegmentDate_ptr);
00029         return;
00030     }
00031     // Retrieve the first and the last legs of the routing.
00032     // Note that in the majority of the cases, as flights are mono-legs,
00033     // the first and last legs are thus the same.
00034     const stdair::LegDateList_T& lLegDateList =
00035         stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00036     stdair::LegDateList_T::const_iterator itFirstLeg = lLegDateList.begin();
00037     const stdair::LegDate* lFirstLeg_ptr = *itFirstLeg;
00038     assert (lFirstLeg_ptr != NULL);
00039     stdair::LegDateList_T::const_reverse_iterator itLastLeg =
00040         lLegDateList.rbegin();
00041     const stdair::LegDate* lLastLeg_ptr = *itLastLeg;
00042     assert (lLastLeg_ptr != NULL);
00043
00044     // Set the Boarding Date
00045     const stdair::Date_T& lBoardingDate = lFirstLeg_ptr->getBoardingDate();
00046     ioSegmentDate.setBoardingDate (lBoardingDate);
00047     // Set the Boarding Time
00048     const stdair::Duration_T& lBoardingTime = lFirstLeg_ptr->getBoardingTime();
00049     ioSegmentDate.setBoardingTime (lBoardingTime);
00050     // Set the Off Date
00051     const stdair::Date_T& lOffDate = lLastLeg_ptr->getOffDate();
00052     ioSegmentDate.setOffDate (lOffDate);
00053     // Set the Off Time
00054     const stdair::Duration_T& lOffTime = lLastLeg_ptr->getOffTime();
00055     ioSegmentDate.setOffTime (lOffTime);
00056     // Set the Elapsed Time for the whole path
00057     updateElapsedTimeFromRouting (ioSegmentDate);
00058
00059     // Initialise the AU for all classes.
00060     const stdair::SegmentCabinList_T& lSegmentCabinList =
00061         stdair::BomManager::getList<stdair::SegmentCabin> (ioSegmentDate);
00062     for (stdair::SegmentCabinList_T::const_iterator itSC =
00063         lSegmentCabinList.begin(); itSC != lSegmentCabinList.end(); ++itSC)
00064     {
00065         stdair::SegmentCabin* lSC_ptr = *itSC;
00066         assert (lSC_ptr != NULL);
00067
00068         // Initialise the AU for children booking classes.
00069         SegmentCabinHelper::initialiseAU (*lSC_ptr);
00070     }
00071 }
00072 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00073 void SegmentDateHelper::
00074 updateElapsedTimeFromRouting (stdair::SegmentDate& ioSegmentDate) {
00075     const stdair::LegDateList_T& lLegDateList =
00076         stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00077
00078     stdair::LegDateList_T::const_iterator itLegDate = lLegDateList.begin();
00079     const stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00080     assert (lCurrentLegDate_ptr != NULL);
00081
00082     // Retrieve the elapsed time of the first leg
00083     stdair::Duration_T lElapsedTime = lCurrentLegDate_ptr->getElapsedTime();
00084
00085     // Go to the next leg, if existing. If not existing, the following
00086     // loop will not be entered (as it means: currentLeg ==
00087     _legDateList.end()).
00088     ++itLegDate;
00089
00090     for (const stdair::LegDate* lPreviousLegDate_ptr = lCurrentLegDate_ptr;
00091         itLegDate != lLegDateList.end();
00092         ++itLegDate, lPreviousLegDate_ptr = lCurrentLegDate_ptr) {
00093         lCurrentLegDate_ptr = *itLegDate;
00094
00095         // As the boarding point of the current leg is the same as the off point
00096         // of the previous leg (by construction), there is no time difference.
00097         assert (lCurrentLegDate_ptr->getBoardingPoint()
00098             == lPreviousLegDate_ptr->getOffPoint());
00099         const stdair::Duration_T& lStopOverTime =

```

```

00099         lCurrentLegDate_ptr->getBoardingTime() - lPreviousLegDate_ptr->
getOffTime();
00100         lElapsedTime += lStopOverTime;
00101
00102         // Add the elapsed time of the current leg
00103         const stdair::Duration_T& currentElapsedTime =
00104         lCurrentLegDate_ptr->getElapsedTime();
00105         lElapsedTime += currentElapsedTime;
00106     }
00107
00108     // Store the result
00109     ioSegmentDate.setElapsedTime (lElapsedTime);
00110     // From the elapsed time, update the distance
00111     updateDistanceFromElapsedTime (ioSegmentDate);
00112 }
00113
00114 // //////////////////////////////////////
00115 void SegmentDateHelper::
00116 updateDistanceFromElapsedTime (stdair::SegmentDate& ioSegmentDate) {
00117     const stdair::Duration_T& lElapsedTime = ioSegmentDate.getElapsedTime();
00118     const double lElapseInHours=static_cast<const double>(lElapsedTime.hours())
;
00119     const long int lDistance =
00120         static_cast<const long int>(stdair::DEFAULT_FLIGHT_SPEED*lElapseInHours);
00121     ioSegmentDate.setDistance (lDistance);
00122 }
00123
00124 }

```

25.103 airinv/bom/SegmentDateHelper.hpp File Reference

Classes

- class [AIRINV::SegmentDateHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.104 SegmentDateHelper.hpp

```

00001 #ifndef __AIRINV_BOM_SEGMENTDATEHELPER_HPP
00002 #define __AIRINV_BOM_SEGMENTDATEHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 // Forward declarations
00009 namespace stdair {
00010     class SegmentDate;
00011 }
00012
00013 namespace AIRINV {
00016     class SegmentDateHelper {
00017     public:
00018         // ////////////////////////////////// Business Methods //////////////////////////////////
00021         static void fillFromRouting (stdair::SegmentDate&);
00022
00032         static void updateElapsedTimeFromRouting (stdair::SegmentDate&);
00033
00035         static void updateDistanceFromElapsedTime (stdair::SegmentDate&);
00036     };
00037
00038 }
00039 #endif // __AIRINV_BOM_SEGMENTDATEHELPER_HPP

```

25.105 airinv/bom/SegmentStruct.cpp File Reference

```
#include <cassert>
```

```
#include <stdair/bom/SegmentDate.hpp>
#include <airinv/bom/SegmentStruct.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.106 SegmentStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/SegmentDate.hpp>
00008 // AIRINV
00009 #include <airinv/bom/SegmentStruct.hpp>
00010
00011 namespace AIRINV {
00012
00013 // //////////////////////////////////////
00014 const std::string SegmentStruct::describe() const {
00015     std::ostringstream ostr;
00016
00017     ostr << "      " << _boardingPoint << " / "
00018         << boost::posix_time::to_simple_string(_boardingTime)
00019         << " -- " << _offPoint << " / "
00020         << boost::posix_time::to_simple_string(_offTime)
00021         << " --> "
00022         << boost::posix_time::to_simple_string(_elapsed)
00023         << std::endl;
00024
00025     for (SegmentCabinStructList_T::const_iterator itCabin =
00026         _cabinList.begin(); itCabin != _cabinList.end(); itCabin++) {
00027         const SegmentCabinStruct& lCabin = *itCabin;
00028         ostr << lCabin.describe();
00029     }
00030     ostr << std::endl;
00031
00032     return ostr.str();
00033 }
00034
00035 // //////////////////////////////////////
00036 void SegmentStruct::fill (stdair::SegmentDate& ioSegmentDate) const {
00037     // Set the Boarding Date
00038     ioSegmentDate.setBoardingDate (_offDate);
00039     // Set the Boarding Time
00040     ioSegmentDate.setBoardingTime (_boardingTime);
00041     // Set the Off Date
00042     ioSegmentDate.setOffDate (_offDate);
00043     // Set the Off Time
00044     ioSegmentDate.setOffTime (_offTime);
00045     // Set the Elapsed Time
00046     ioSegmentDate.setElapsedTime (_elapsed);
00047 }
00048
00049 }
```

25.107 airinv/bom/SegmentStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
```

Classes

- struct [AIRINV::SegmentStruct](#)

Namespaces

- namespace `stdair`
Forward declarations.
- namespace `AIRINV`

Typedefs

- typedef `std::vector`
 < SegmentStruct > `AIRINV::SegmentStructList_T`

25.108 SegmentStruct.hpp

```

00001 #ifndef __AIRINV_BOM_SEGMENTSTRUCT_HPP
00002 #define __AIRINV_BOM_SEGMENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AIRINV
00014 #include <airinv/bom/SegmentCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentDate;
00019 }
00020
00021 namespace AIRINV {
00022     struct SegmentStruct : public stdair::StructAbstract {
00023         // Attributes
00024         stdair::AirportCode_T _boardingPoint;
00025         stdair::AirportCode_T _offPoint;
00026         stdair::Date_T _boardingDate;
00027         stdair::Duration_T _boardingTime;
00028         stdair::Date_T _offDate;
00029         stdair::Duration_T _offTime;
00030         stdair::Duration_T _elapsed;
00031         SegmentCabinStructList_T _cabinList;
00032
00033         void fill (stdair::SegmentDate&) const;
00034
00035         const std::string describe() const;
00036     };
00037
00038     typedef std::vector<SegmentStruct> SegmentStructList_T;
00039 }
00040
00041 #endif // __AIRINV_BOM_SEGMENTSTRUCT_HPP

```

25.109 airinv/command/InventoryBuilder.cpp File Reference

```
#include <cassert>
```

```

#include <boost/date_time/date_iterator.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/FlightDateStruct.hpp>
#include <airinv/command/InventoryBuilder.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.110 InventoryBuilder.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/date_time/date_iterator.hpp>
00008 // StdAir
00009 #include <stdair/basic/BasConst_BookingClass.hpp>
00010 #include <stdair/basic/BasConst_Yield.hpp>
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/Inventory.hpp>
00015 #include <stdair/bom/FlightDate.hpp>
00016 #include <stdair/bom/SegmentDate.hpp>
00017 #include <stdair/bom/SegmentCabin.hpp>
00018 #include <stdair/bom/FareFamily.hpp>
00019 #include <stdair/bom/BookingClass.hpp>
00020 #include <stdair/bom/LegDate.hpp>
00021 #include <stdair/bom/LegCabin.hpp>
00022 #include <stdair/bom/Bucket.hpp>
00023 #include <stdair/factory/FacBom.hpp>
00024 #include <stdair/factory/FacBomManager.hpp>
00025 #include <stdair/service/Logger.hpp>
00026 // AirInv
00027 #include <airinv/bom/FlightDateStruct.hpp>
00028 #include <airinv/command/InventoryBuilder.hpp>
00029
00030 namespace AIRINV {
00031
00032 // //////////////////////////////////////
00033 void InventoryBuilder::
00034 buildInventory (stdair::BomRoot& ioBomRoot,
00035                const FlightDateStruct& iFlightDateStruct) {
00036     const stdair::AirlineCode_T& lAirlineCode = iFlightDateStruct._airlineCode;
00037
00038     // Instantiate an inventory object (if not exist)
00039     // for the given key (airline code)
00040     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00041         getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00042     if (lInventory_ptr == NULL) {
00043         stdair::InventoryKey lKey (lAirlineCode);

```

```

00044     lInventory_ptr =
00045         &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00046     stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00047     stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00048 }
00049 assert (lInventory_ptr != NULL);
00050
00051 // Build the flight-date within the inventory.
00052 buildFlightDate (*lInventory_ptr, iFlightDateStruct);
00053 }
00054
00055 // //////////////////////////////////////
00056 void InventoryBuilder::
00057 buildFlightDate (stdair::Inventory& ioInventory,
00058                 const FlightDateStruct& iFlightDateStruct) {
00059     // Create the FlightDateKey
00060     const stdair::FlightDateKey lFlightDateKey (iFlightDateStruct._flightNumber
00061
00062                                     iFlightDateStruct._flightDate);
00063
00064     // Check that the flight-date object is not already existing. If a
00065     // flight-date object with the same key has already been created,
00066     // then just update it, ifnot, create a flight-date and update it.
00067     stdair::FlightDate* lFlightDate_ptr = stdair::BomManager::
00068         getObjectPtr<stdair::FlightDate> (ioInventory, lFlightDateKey.toString())
00069 ;
00070     if (lFlightDate_ptr == NULL) {
00071         // Instantiate a flighty-date object for the given key (flight number and
00072         // flight date)
00073         lFlightDate_ptr =
00074             &stdair::FacBom<stdair::FlightDate>::instance().create (lFlightDateKey)
00075 ;
00076         stdair::FacBomManager::addToListAndMap (ioInventory, *lFlightDate_ptr);
00077         stdair::FacBomManager::linkWithParent (ioInventory, *lFlightDate_ptr);
00078     }
00079     assert (lFlightDate_ptr != NULL);
00080
00081     // Update the BOM flight-date with the attributes of the flight-date
00082     struct.
00083
00084     // Browse the list of leg-date struct and segment-date struct and
00085     // create the corresponding BOM.
00086     for (LegStructList_T::const_iterator itLegDate =
00087         iFlightDateStruct._legList.begin();
00088         itLegDate != iFlightDateStruct._legList.end(); ++itLegDate) {
00089         const LegStruct& lCurrentLegDateStruct = *itLegDate;
00090         buildLegDate (*lFlightDate_ptr, lCurrentLegDateStruct);
00091     }
00092
00093     for (SegmentStructList_T::const_iterator itSegmentDate =
00094         iFlightDateStruct._segmentList.begin();
00095         itSegmentDate != iFlightDateStruct._segmentList.end();
00096         ++itSegmentDate) {
00097         const SegmentStruct& lCurrentSegmentDateStruct = *itSegmentDate;
00098         buildSegmentDate (*lFlightDate_ptr, lCurrentSegmentDateStruct);
00099     }
00100 }
00101
00102 // //////////////////////////////////////
00103 void InventoryBuilder::
00104 buildLegDate (stdair::FlightDate& ioFlightDate,
00105              const LegStruct& iLegDateStruct) {
00106     // Check that the leg-date object is not already existing. If a
00107     // leg-date object with the same key has already been created,
00108     // then just update it, ifnot, create a leg-date and update it.
00109     stdair::LegDate* lLegDate_ptr = stdair::BomManager::
00110         getObjectPtr<stdair::LegDate> (ioFlightDate, iLegDateStruct._boardingPoint
00111 );
00112     if (lLegDate_ptr == NULL) {
00113         // Instantiate a leg-date object for the given key (boarding point);
00114         stdair::LegDateKey lKey (iLegDateStruct._boardingPoint);
00115         lLegDate_ptr = &stdair::FacBom<stdair::LegDate>::instance().create (lKey)
00116 ;
00117         stdair::FacBomManager::addToListAndMap (ioFlightDate, *lLegDate_ptr);
00118         stdair::FacBomManager::linkWithParent (ioFlightDate, *lLegDate_ptr);
00119     }
00120     assert (lLegDate_ptr != NULL);
00121
00122     // Update the BOM leg-date with the attributes of the leg-date struct.
00123     iLegDateStruct.fill (*lLegDate_ptr);
00124
00125     // Browse the list of leg-cabin structs and create the corresponding BOM.
00126     for (LegCabinStructList_T::const_iterator itLegCabin =
00127         iLegDateStruct._cabinList.begin();
00128         itLegCabin != iLegDateStruct._cabinList.end(); ++itLegCabin) {
00129         const LegCabinStruct& lCurrentLegCabinStruct = *itLegCabin;

```

```

00125     buildLegCabin (*lLegDate_ptr, lCurrentLegCabinStruct);
00126 }
00127 }
00128
00129 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00130 void InventoryBuilder::
00131 buildLegCabin (stdair::LegDate& ioLegDate,
00132               const LegCabinStruct& iLegCabinStruct) {
00133     // Check that the leg-cabin object is not already existing. If a
00134     // leg-cabin object with the same key has already been created,
00135     // then just update it, ifnot, create a leg-cabin and update it.
00136     stdair::LegCabin* lLegCabin_ptr = stdair::BomManager::
00137         getObjectPtr<stdair::LegCabin> (ioLegDate, iLegCabinStruct._cabinCode);
00138     if (lLegCabin_ptr == NULL) {
00139         // Instantiate a leg-cabin object for the given key (cabin code);
00140         stdair::LegCabinKey lKey (iLegCabinStruct._cabinCode);
00141         lLegCabin_ptr = &stdair::FacBom<stdair::LegCabin>::instance().create(lKey
00142 );
00143         stdair::FacBomManager::addToListAndMap (ioLegDate, *lLegCabin_ptr);
00144         stdair::FacBomManager::linkWithParent (ioLegDate, *lLegCabin_ptr);
00145     }
00146     assert (lLegCabin_ptr != NULL);
00147
00148     // TODO: Update the BOM leg-cabin with the attributes of the
00149     // leg-cabin struct.
00150     iLegCabinStruct.fill (*lLegCabin_ptr);
00151
00152     // Browse the list of bucket structs and create the corresponding BOM.
00153     for (BucketStructList_T::const_iterator itBucket =
00154         iLegCabinStruct._bucketList.begin();
00155         itBucket != iLegCabinStruct._bucketList.end(); ++itBucket) {
00156         const BucketStruct& lCurrentBucketStruct = *itBucket;
00157         buildBucket (*lLegCabin_ptr, lCurrentBucketStruct);
00158     }
00159 }
00160 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00161 void InventoryBuilder::buildBucket (stdair::LegCabin& ioLegCabin,
00162                                    const BucketStruct& iBucketStruct) {
00163     // Create the BucketKey
00164     const stdair::BucketKey lBucketKey (iBucketStruct._seatIndex);
00165
00166     // Check that the bucket object is not already existing. If a
00167     // bucket object with the same key has already been created,
00168     // then just update it, ifnot, create a bucket and update it.
00169     stdair::Bucket* lBucket_ptr = stdair::BomManager::
00170         getObjectPtr<stdair::Bucket> (ioLegCabin, lBucketKey.toString());
00171     if (lBucket_ptr == NULL) {
00172         // Instantiate a bucket object for the given key (seat index);
00173         stdair::BucketKey lKey (iBucketStruct._seatIndex);
00174         lBucket_ptr = &stdair::FacBom<stdair::Bucket>::instance().create (lKey);
00175         stdair::FacBomManager::addToListAndMap (ioLegCabin, *lBucket_ptr);
00176         stdair::FacBomManager::linkWithParent (ioLegCabin, *lBucket_ptr);
00177     }
00178     assert (lBucket_ptr != NULL);
00179
00180     //
00181     iBucketStruct.fill (*lBucket_ptr);
00182 }
00183
00184 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00185 void InventoryBuilder::
00186 buildSegmentDate (stdair::FlightDate& ioFlightDate,
00187                  const SegmentStruct& iSegmentDateStruct) {
00188     // Check that the segment-date object is not already existing. If a
00189     // segment-date object with the same key has already been created,
00190     // then just update it, ifnot, create a segment-date and update it.
00191     const stdair::SegmentDateKey
00192         lSegmentDateKey (iSegmentDateStruct._boardingPoint,
00193                         iSegmentDateStruct._offPoint);
00194     stdair::SegmentDate* lSegmentDate_ptr = stdair::BomManager::
00195         getObjectPtr<stdair::SegmentDate> (ioFlightDate, lSegmentDateKey.toString()
00196 );
00197     if (lSegmentDate_ptr == NULL) {
00198         // Instantiate a segment-date object for the given key (boarding
00199         // and off points);
00200         lSegmentDate_ptr = &stdair::FacBom<stdair::SegmentDate>::
00201             instance().create (lSegmentDateKey);
00202         stdair::FacBomManager::addToListAndMap (ioFlightDate, *lSegmentDate_ptr);
00203         stdair::FacBomManager::linkWithParent (ioFlightDate, *lSegmentDate_ptr);
00204     }
00205     assert (lSegmentDate_ptr != NULL);
00206
00207     // Update the BOM segment-date with the attributes of the
00208     // segment-date struct.
00209     iSegmentDateStruct.fill (*lSegmentDate_ptr);

```

```

00210 // Browse the list of segment-cabin struct and create the corresponding
00211 BOM.
00212 for (SegmentCabinStructList_T::const_iterator itSegmentCabin =
00213     iSegmentDateStruct._cabinList.begin();
00214     itSegmentCabin != iSegmentDateStruct._cabinList.end();
00215     ++itSegmentCabin) {
00216     const SegmentCabinStruct& lCurrentSegmentCabinStruct = *itSegmentCabin;
00217     buildSegmentCabin (*lSegmentDate_ptr, lCurrentSegmentCabinStruct);
00218 }
00219
00220 // //////////////////////////////////////
00221 void InventoryBuilder::
00222 buildSegmentCabin (stdair::SegmentDate& ioSegmentDate,
00223     const SegmentCabinStruct& iSegmentCabinStruct) {
00224     // Check that the segment-cabin object is not already existing. If a
00225     // segment-cabin object with the same key has already been created,
00226     // then just update it, ifnot, create a segment-cabin and update it.
00227     stdair::SegmentCabin* lSegmentCabin_ptr = stdair::BomManager::
00228         getObjectPtr<stdair::SegmentCabin> (ioSegmentDate,
00229             iSegmentCabinStruct._cabinCode);
00230     if (lSegmentCabin_ptr == NULL) {
00231         // Instantiate a segment-cabin object for the given key (cabin code);
00232         stdair::SegmentCabinKey lKey (iSegmentCabinStruct._cabinCode);
00233         lSegmentCabin_ptr =
00234             &stdair::FacBom<stdair::SegmentCabin>::instance().create (lKey);
00235     }
00236     // Link the segment-cabin to the segment-date
00237     stdair::FacBomManager::addToListAndMap (ioSegmentDate, *lSegmentCabin_ptr);
00238
00239     stdair::FacBomManager::linkWithParent (ioSegmentDate, *lSegmentCabin_ptr);
00240
00241 }
00242 assert (lSegmentCabin_ptr != NULL);
00243
00244 // TODO: Update the BOM segment-cabin with the attributes of the
00245 // segment-cabin struct.
00246 iSegmentCabinStruct.fill (*lSegmentCabin_ptr);
00247
00248 // Browse the list of fare family struct and create the corresponding BOM.
00249 for (FareFamilyStructList_T::const_iterator itFareFamily =
00250     iSegmentCabinStruct._fareFamilies.begin();
00251     itFareFamily != iSegmentCabinStruct._fareFamilies.end();
00252     ++itFareFamily) {
00253     const FareFamilyStruct& lCurrentFareFamilyStruct = *itFareFamily;
00254     buildFareFamily (*lSegmentCabin_ptr, lCurrentFareFamilyStruct);
00255 }
00256
00257 // //////////////////////////////////////
00258 void InventoryBuilder::
00259 buildFareFamily (stdair::SegmentCabin& ioSegmentCabin,
00260     const FareFamilyStruct& iFareFamilyStruct) {
00261     // Check that the fare family object is not already existing. If a
00262     // fare family object with the same key has already been created,
00263     // then just update it. If not, create a fare family and update it.
00264     stdair::FareFamily* lFareFamily_ptr = stdair::BomManager::
00265         getObjectPtr<stdair::FareFamily> (ioSegmentCabin,
00266             iFareFamilyStruct._familyCode);
00267     if (lFareFamily_ptr == NULL) {
00268         // Instantiate a fare family object for the given key (fare family code);
00269         const stdair::FareFamilyKey lFFKey (iFareFamilyStruct._familyCode);
00270         lFareFamily_ptr =
00271             &stdair::FacBom<stdair::FareFamily>::instance().create (lFFKey);
00272     }
00273     // Link the fare family to the segment-cabin
00274     stdair::FacBomManager::addToListAndMap (ioSegmentCabin, *lFareFamily_ptr);
00275
00276     stdair::FacBomManager::linkWithParent (ioSegmentCabin, *lFareFamily_ptr);
00277
00278 }
00279 assert (lFareFamily_ptr != NULL);
00280
00281 // TODO: Upcabin the BOM fare family with the attributes of the
00282 // fare family struct.
00283 iFareFamilyStruct.fill (*lFareFamily_ptr);
00284
00285 // Browse the list of booking-class struct and create the corresponding
00286 BOM.
00287 for (BookingClassStructList_T::const_iterator itBookingClass =
00288     iFareFamilyStruct._classList.begin();
00289     itBookingClass != iFareFamilyStruct._classList.end();
00290     ++itBookingClass) {
00291     const BookingClassStruct& lCurrentBookingClassStruct = *itBookingClass;
00292     buildBookingClass (*lFareFamily_ptr, lCurrentBookingClassStruct);
00293 }
00294
00295 }

```



```

00292
00293 ///////////////////////////////////////////////////////////////////
00294 void InventoryBuilder::
00295 buildBookingClass (stdair::FareFamily& ioFareFamily,
00296                   const BookingClassStruct& iBookingClassStruct) {
00297
00298     // Check that the booking class object is not already existing. If a
00299     // booking-class object with the same key has already been created,
00300     // then just update it. If not, create a booking-class and update it.
00301     stdair::BookingClass* lBookingClass_ptr = stdair::BomManager::
00302         getObjectPtr<stdair::BookingClass> (ioFareFamily,
00303                                             iBookingClassStruct._classCode);
00304     if (lBookingClass_ptr == NULL) {
00305         // Instantiate a booking class object for the given key (class code);
00306         const stdair::BookingClassKey lClassKey (iBookingClassStruct._classCode);
00307         lBookingClass_ptr =
00308             &stdair::FacBom<stdair::BookingClass>::instance().create (lClassKey);
00309
00310         // Link the booking-class to the fare family
00311         stdair::FacBomManager::addToListAndMap (ioFareFamily, *lBookingClass_ptr)
00312     ;
00313
00314         // Link the booking-class to the segment-cabin
00315         stdair::SegmentCabin& lSegmentCabin =
00316             stdair::BomManager::getParent<stdair::SegmentCabin> (ioFareFamily);
00317         stdair::FacBomManager::addToListAndMap (lSegmentCabin, *lBookingClass_ptr)
00318     };
00319
00320         // Link the booking-class to the segment-date
00321         stdair::SegmentDate& lSegmentDate =
00322             stdair::BomManager::getParent<stdair::SegmentDate> (lSegmentCabin);
00323         stdair::FacBomManager::addToListAndMap (lSegmentDate, *lBookingClass_ptr)
00324     ;
00325     }
00326     assert (lBookingClass_ptr != NULL);
00327
00328     // TODO: Upcabin the BOM booking-class with the attributes of the
00329     // booking-class struct.
00330     iBookingClassStruct.fill (*lBookingClass_ptr);
00331 }

```

25.111 airinv/command/InventoryBuilder.hpp File Reference

```

#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- class [AIRINV::InventoryBuilder](#)
Class handling the generation / instantiation of the Inventory BOM.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::InventoryParserHelper](#)

25.112 InventoryBuilder.hpp

```

00001 #ifndef __AIRINV_CMD_INVENTORYBUILDER_HPP
00002 #define __AIRINV_CMD_INVENTORYBUILDER_HPP
00003
00004 ///////////////////////////////////////////////////////////////////
00005 // Import section
00006 ///////////////////////////////////////////////////////////////////
00007 // StdAir

```

```

00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirInv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00013 namespace stdair {
00014     class BomRoot;
00015     class Inventory;
00016     class FlightDate;
00017     class LegDate;
00018     class LegCabin;
00019     class Bucket;
00020     class SegmentDate;
00021     class SegmentCabin;
00022     class FareFamily;
00023 }
00024
00025 namespace AIRINV {
00026
00028     struct FlightDateStruct;
00029     struct LegStruct;
00030     struct LegCabinStruct;
00031     struct BucketStruct;
00032     struct SegmentStruct;
00033     struct SegmentCabinStruct;
00034     struct FareFamilyStruct;
00035     struct BookingClassStruct;
00036     namespace InventoryParserHelper {
00037         struct doEndFlightDate;
00038     }
00039
00043     class InventoryBuilder : public stdair::CmdAbstract {
00049         friend struct InventoryParserHelper::doEndFlightDate;
00050
00051     private:
00056         static void buildInventory (stdair::BomRoot&, const FlightDateStruct&);
00057
00062         static void buildFlightDate (stdair::Inventory&, const FlightDateStruct&);
00063
00068         static void buildLegDate (stdair::FlightDate&, const LegStruct&);
00069
00074         static void buildLegCabin (stdair::LegDate&, const LegCabinStruct&);
00075
00080         static void buildBucket (stdair::LegCabin&, const BucketStruct&);
00081
00086         static void buildSegmentDate (stdair::FlightDate&, const SegmentStruct&);
00087
00092         static void buildSegmentCabin (stdair::SegmentDate&,
00093                                         const SegmentCabinStruct&);
00094
00099         static void buildFareFamily (stdair::SegmentCabin&,
00100                                     const FareFamilyStruct&);
00101
00106         static void buildBookingClass (stdair::FareFamily&,
00107                                       const BookingClassStruct&);
00108     };
00109
00110 }
00111 #endif // __AIRINV_CMD_INVENTORYBUILDER_HPP

```

25.113 airinv/command/InventoryGenerator.cpp File Reference

```
#include <cassert>
```

```

#include <boost/date_time/date_iterator.hpp>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>
#include <airinv/command/InventoryGenerator.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.114 InventoryGenerator.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/date_time/date_iterator.hpp>
00008 // StdAir
00009 #include <stdair/stdair_types.hpp>
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/FlightDate.hpp>
00015 #include <stdair/bom/SegmentDate.hpp>
00016 #include <stdair/bom/SegmentCabin.hpp>
00017 #include <stdair/bom/FareFamily.hpp>
00018 #include <stdair/bom/BookingClass.hpp>
00019 #include <stdair/bom/LegDate.hpp>
00020 #include <stdair/bom/LegCabin.hpp>
00021 #include <stdair/bom/Bucket.hpp>
00022 #include <stdair/factory/FacBomManager.hpp>
00023 #include <stdair/service/Logger.hpp>
00024 // AirInv
00025 #include <airinv/bom/FlightPeriodStruct.hpp>
00026 #include <airinv/command/InventoryGenerator.hpp>
00027
00028 namespace AIRINV {
00029
00030 // //////////////////////////////////////
00031 void InventoryGenerator::
00032 createFlightDate (stdair::BomRoot& ioBomRoot,
00033                  const FlightPeriodStruct& iFlightPeriod) {
00034     const stdair::AirlineCode_T& lAirlineCode = iFlightPeriod._airlineCode;
00035
00036     // Instantiate an inventory object (if not exist)
00037     // for the given key (airline code)
00038     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00039         getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00040     if (lInventory_ptr == NULL) {
00041         stdair::InventoryKey lKey (lAirlineCode);
00042         lInventory_ptr =
00043             &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00044         stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00045         stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00046     }

```

```

00047     assert (lInventory_ptr != NULL);
00048
00049     // Generate all the dates corresponding to the period
00050     // and create the corresponding flight-dates.
00051     const stdair::DatePeriod_T lDateRange = iFlightPeriod._dateRange;
00052
00053     for (boost::gregorian::day_iterator itDate = lDateRange.begin();
00054          itDate != lDateRange.end(); ++itDate) {
00055         const stdair::Date_T& currentDate = *itDate;
00056
00057         // Retrieve, for the current day, the Day-Of-the-Week (thanks to Boost)
00058         const unsigned short currentDoW = currentDate.day_of_week().as_number();
00059
00060         // The FlightPeriod structure stores which Days (-Of-the-Week) are
00061         // active within the week. For each day (Mon., Tue., etc.), a boolean
00062         // states whether the Flight is active for that day.
00063         const stdair::DoWStruct& lDoWList = iFlightPeriod._dow;
00064         const bool isDoWActive = lDoWList.getStandardDayOfWeek (currentDoW);
00065
00066         if (isDoWActive == true) {
00067             createFlightDate (*lInventory_ptr, currentDate, iFlightPeriod);
00068         }
00069     }
00070 }
00071
00072 // //////////////////////////////////////
00073 void InventoryGenerator::
00074 createFlightDate (stdair::Inventory& ioInventory,
00075                  const stdair::Date_T& iFlightDate,
00076                  const FlightPeriodStruct& iFlightPeriod) {
00077     // Create the FlightDateKey
00078     const stdair::FlightNumber_T& lFlightNumber = iFlightPeriod._flightNumber;
00079     stdair::FlightDateKey lFlightDateKey (lFlightNumber, iFlightDate);
00080
00081     // DEBUG
00082     // STDAIR_LOG_DEBUG ("Creating flight-date: " <<
lFlightDateKey.toString());
00083
00084     // Check that the flight-date object is not already existing. If a
00085     // FlightDate object with the same key has already been created,
00086     // it means that the schedule input file is invalid (two flight-periods
00087     // are overlapping).
00088     stdair::FlightDate* lFlightDate_ptr = stdair::BomManager::
00089         getObjectPtr<stdair::FlightDate> (ioInventory, lFlightDateKey.toString())
;
00090     if (lFlightDate_ptr != NULL) {
00091         std::ostringstream oMessage;
00092         oMessage << ioInventory.describeKey() << ", "
00093             << lFlightDate_ptr->describeKey();
00094         throw FlightDateDuplicationException (oMessage.str());
00095     }
00096
00097     // Instantiate a flight-date object with the given key (flight number and
00098     // flight date)
00099     lFlightDate_ptr =
00100         &stdair::FacBom<stdair::FlightDate>::instance().create (lFlightDateKey);
00101     stdair::FacBomManager::addToListAndMap (ioInventory, *lFlightDate_ptr);
00102     stdair::FacBomManager::linkWithParent (ioInventory, *lFlightDate_ptr);
00103
00104     // Iterate on the leg-dates
00105     stdair::Duration_T currentOffTime (0, 0, 0);
00106     stdair::AirportCode_T previousOffPoint;
00107     const LegStructList_T& lLegList = iFlightPeriod._legList;
00108     for (LegStructList_T::const_iterator itLeg = lLegList.begin();
00109          itLeg != lLegList.end(); ++itLeg) {
00110         const LegStruct& lLeg = *itLeg;
00111
00112         // Create the leg-branch of the flight-date BOM
00113         stdair::LegDate& lLegDate =
00114             createLegDate (*lFlightDate_ptr, iFlightDate, lLeg);
00115
00116         // TODO: Check that the boarding date/time of the next leg is greater
00117         // than the off date/time of the current leg. Throw an exception
00118         // otherwise.
00119
00120         // TODO: specify, in the schedule input file specifications, that the
00121         // legs should be given in their natural order.
00122         // Then, replace the assertion by a thrown exception.
00123         //
00124         // Check that the legs are given in their natural order. If the schedule
00125         // input does not respect that assumption, the following assertion will
00126         // fail.
00127         if (itLeg != lLegList.begin()) {
00128             const stdair::AirportCode_T& currentBoardingPoint =
00129                 lLegDate.getBoardingPoint();
00130             assert (currentBoardingPoint == previousOffPoint);
00131         }

```

```

00132
00133     // Set the local variable for the next iteration
00134     previousOffPoint = lLegDate.getOffPoint();
00135 }
00136
00137 // Iterate on the segment structures
00138 const SegmentStructList_T& lSegmentList = iFlightPeriod._segmentList;
00139 for (SegmentStructList_T::const_iterator itSegment = lSegmentList.begin();
00140      itSegment != lSegmentList.end(); ++itSegment) {
00141     const SegmentStruct& lSegment = *itSegment;
00142
00143     createSegmentDate (*lFlightDate_ptr, lSegment);
00144 }
00145 }
00146
00147 // //////////////////////////////////////
00148 stdair::LegDate& InventoryGenerator::
00149 createLegDate (stdair::FlightDate& ioFlightDate,
00150               const stdair::Date_T& iReferenceDate,
00151               const LegStruct& iLeg) {
00152     // Create the leg-date corresponding to the boarding point.
00153     stdair::LegDateKey lKey (iLeg._boardingPoint);
00154     stdair::LegDate& lLegDate =
00155         stdair::FacBom<stdair::LegDate>::instance().create (lKey);
00156     stdair::FacBomManager::addToListAndMap (ioFlightDate, lLegDate);
00157     stdair::FacBomManager::linkWithParent (ioFlightDate, lLegDate);
00158
00159     // Set the leg-date attributes
00160     iLeg.fill (iReferenceDate, lLegDate);
00161
00162     // Iterate on the cabins
00163     const LegCabinStructList_T& lCabinList = iLeg._cabinList;
00164     for (LegCabinStructList_T::const_iterator itCabin = lCabinList.begin();
00165          itCabin != lCabinList.end(); ++itCabin) {
00166         const LegCabinStruct& lCabin = *itCabin;
00167
00168         // Create the leg-cabin-branch of the leg-date
00169         createLegCabin (lLegDate, lCabin);
00170     }
00171
00172     return lLegDate;
00173 }
00174
00175 // //////////////////////////////////////
00176 void InventoryGenerator::
00177 createLegCabin (stdair::LegDate& ioLegDate,
00178                const LegCabinStruct& iCabin) {
00179     // Instantiate an leg-cabin object with the corresponding cabin code
00180     const stdair::LegCabinKey lKey (iCabin._cabinCode);
00181     stdair::LegCabin& lLegCabin =
00182         stdair::FacBom<stdair::LegCabin>::instance().create (lKey);
00183     stdair::FacBomManager::addToListAndMap (ioLegDate, lLegCabin);
00184     stdair::FacBomManager::linkWithParent (ioLegDate, lLegCabin);
00185
00186     // Set the Leg-Cabin attributes
00187     iCabin.fill (lLegCabin);
00188
00189     // Iterate on the bucket
00190     const BucketStructList_T& lBucketList = iCabin._bucketList;
00191     for (BucketStructList_T::const_iterator itBucket = lBucketList.begin();
00192          itBucket != lBucketList.end(); ++itBucket) {
00193         const BucketStruct& lBucket = *itBucket;
00194
00195         // Create the bucket of the leg-cabin
00196         createBucket (lLegCabin, lBucket);
00197     }
00198 }
00199
00200 // //////////////////////////////////////
00201 void InventoryGenerator::createBucket (stdair::LegCabin& ioLegCabin,
00202                                     const BucketStruct& iBucket) {
00203     // Instantiate a bucket object with the corresponding seat index
00204     const stdair::BucketKey lKey (iBucket._seatIndex);
00205     stdair::Bucket& lBucket =
00206         stdair::FacBom<stdair::Bucket>::instance().create (lKey);
00207     stdair::FacBomManager::addToListAndMap (ioLegCabin, lBucket);
00208     stdair::FacBomManager::linkWithParent (ioLegCabin, lBucket);
00209
00210     // Set the Bucket attributes
00211     iBucket.fill (lBucket);
00212 }
00213
00214 // //////////////////////////////////////
00215 void InventoryGenerator::
00216 createSegmentDate (stdair::FlightDate& ioFlightDate,
00217                  const SegmentStruct& iSegment) {
00218     // Set the segment-date primary key

```

```

00219     const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00220     const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00221     stdair::SegmentDateKey lSegmentDateKey (lBoardingPoint, lOffPoint);
00222     // Instantiate an segment-date object with the key.
00223     stdair::SegmentDate& lSegmentDate =
00224         stdair::FacBom<stdair::SegmentDate>::instance().create (lSegmentDateKey);
00225     stdair::FacBomManager::addToListAndMap (ioFlightDate, lSegmentDate);
00226     stdair::FacBomManager::linkWithParent (ioFlightDate, lSegmentDate);
00227
00228     // Set the segment-date attributes
00229     iSegment.fill (lSegmentDate);
00230
00231     // Iterate on the Cabins
00232     const SegmentCabinStructList_T& lCabinList = iSegment._cabinList;
00233     for (SegmentCabinStructList_T::const_iterator itCabin =
00234         lCabinList.begin(); itCabin != lCabinList.end(); ++itCabin) {
00235         const SegmentCabinStruct& lCabin = *itCabin;
00236
00237         // Create the segment-cabin-branch of the segment-date BOM
00238         createSegmentCabin (lSegmentDate, lCabin);
00239     }
00240 }
00241
00242 // //////////////////////////////////////
00243 void InventoryGenerator::
00244 createSegmentCabin (stdair::SegmentDate& ioSegmentDate,
00245                     const SegmentCabinStruct& iCabin) {
00246
00247     // Instantiate an segment-cabin object with the corresponding cabin code
00248     stdair::SegmentCabinKey lKey (iCabin._cabinCode);
00249     stdair::SegmentCabin& lSegmentCabin =
00250         stdair::FacBom<stdair::SegmentCabin>::instance().create (lKey);
00251
00252     // Link the segment-cabin to its parent, the segment-date
00253     stdair::FacBomManager::addToListAndMap (ioSegmentDate, lSegmentCabin);
00254     stdair::FacBomManager::linkWithParent (ioSegmentDate, lSegmentCabin);
00255
00256     // Set the segment-cabin attributes
00257     iCabin.fill (lSegmentCabin);
00258
00259     // Create the list of fare families
00260     for (FareFamilyStructList_T::const_iterator itFareFamily =
00261         iCabin._fareFamilies.begin();
00262         itFareFamily != iCabin._fareFamilies.end(); itFareFamily++) {
00263         const FareFamilyStruct& lFareFamilyStruct = *itFareFamily;
00264
00265         //
00266         createFareFamily (lSegmentCabin, lFareFamilyStruct);
00267     }
00268 }
00269
00270 // //////////////////////////////////////
00271 void InventoryGenerator::
00272 createFareFamily (stdair::SegmentCabin& ioSegmentCabin,
00273                  const FareFamilyStruct& iFF) {
00274     // Instantiate an segment-cabin object with the corresponding cabin code
00275     stdair::FareFamilyKey lKey (iFF._familyCode);
00276     stdair::FareFamily& lFareFamily =
00277         stdair::FacBom<stdair::FareFamily>::instance().create (lKey);
00278
00279     // Link the fare family to its parent, the segment-cabin
00280     stdair::FacBomManager::addToListAndMap (ioSegmentCabin,
00281                                             lFareFamily);
00282     stdair::FacBomManager::linkWithParent (ioSegmentCabin,
00283                                             lFareFamily);
00284
00285     // Set the fare family attributes
00286     iFF.fill (lFareFamily);
00287
00288     // Iterate on the classes
00289     const stdair::ClassList_String_T& lClassList = iFF._classes;
00290     for (stdair::ClassList_String_T::const_iterator itClass =
00291         lClassList.begin(); itClass != lClassList.end(); ++itClass) {
00292         // Transform the single-character class code into a STL string
00293         std::ostream ostr;
00294         ostr << *itClass;
00295         const stdair::ClassCode_T lClassCode (ostr.str());
00296
00297         // Create the booking class branch of the segment-cabin BOM
00298         createClass (lFareFamily, lClassCode);
00299     }
00300 }
00301
00302 // //////////////////////////////////////
00303 void InventoryGenerator::createClass (stdair::FareFamily& ioFareFamily,
00304                                     const stdair::ClassCode_T& iClassCode)
00305 {

```

```

00305
00306 // Instantiate a booking class object with the given class code
00307 const stdair::BookingClassKey lClassKey (iClassCode);
00308 stdair::BookingClass& lClass =
00309     stdair::FacBom<stdair::BookingClass>::instance().create (lClassKey);
00310
00311 // Link the booking-class to the fare family
00312 stdair::FacBomManager::addToListAndMap (ioFareFamily, lClass);
00313 stdair::FacBomManager::linkWithParent (ioFareFamily, lClass);
00314
00315 // Link the booking-class to the segment-cabin
00316 stdair::SegmentCabin& lSegmentCabin =
00317     stdair::BomManager::getParent<stdair::SegmentCabin> (ioFareFamily);
00318 stdair::FacBomManager::addToListAndMap (lSegmentCabin, lClass);
00319
00320 // Link the booking-class to the segment-date
00321 stdair::SegmentDate& lSegmentDate =
00322     stdair::BomManager::getParent<stdair::SegmentDate> (lSegmentCabin);
00323 stdair::FacBomManager::addToListAndMap (lSegmentDate, lClass);
00324 }
00325 }

```

25.115 airinv/command/InventoryGenerator.hpp File Reference

```

#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- class [AIRINV::InventoryGenerator](#)
Class handling the generation / instantiation of the Inventory BOM.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::ScheduleParserHelper](#)

25.116 InventoryGenerator.hpp

```

00001 #ifndef __AIRINV_CMD_INVENTORYGENERATOR_HPP
00002 #define __AIRINV_CMD_INVENTORYGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00012 namespace stdair {
00013     class BomRoot;
00014     class Inventory;
00015     class FlightDate;
00016     class LegDate;
00017     class LegCabin;
00018     class SegmentDate;
00019     class SegmentCabin;
00020     class FareFamily;
00021 }
00022
00023 namespace AIRINV {
00024     // Forward declarations
00025     struct FlightPeriodStruct;
00026     struct LegStruct;
00027     struct SegmentStruct;
00028     struct LegCabinStruct;

```

```

00031 struct SegmentCabinStruct;
00032 struct FareFamilyStruct;
00033 struct BucketStruct;
00034 namespace ScheduleParserHelper {
00035     struct doEndFlight;
00036 }
00037
00042 class InventoryGenerator : public stdair::CmdAbstract {
00048     friend class FlightPeriodFileParser;
00049     friend class FFFlightPeriodFileParser;
00050     friend struct ScheduleParserHelper::doEndFlight;
00051     friend class ScheduleParser;
00052
00053 private:
00058     static void createFlightDate (stdair::BomRoot&,
00059                                 const FlightPeriodStruct&);
00060
00064     static void createFlightDate (stdair::Inventory&,
00065                                 const stdair::Date_T&,
00066                                 const FlightPeriodStruct&);
00067
00071     static stdair::LegDate& createLegDate (stdair::FlightDate&,
00072                                           const stdair::Date_T&,
00073                                           const LegStruct&);
00074
00078     static void createLegCabin (stdair::LegDate&, const LegCabinStruct&);
00079
00083     static void createBucket (stdair::LegCabin&, const BucketStruct&);
00084
00088     static void createSegmentDate (stdair::FlightDate&,
00089                                   const SegmentStruct&);
00090
00094     static void createSegmentCabin (stdair::SegmentDate&,
00095                                   const SegmentCabinStruct&);
00096
00100     static void createFareFamily (stdair::SegmentCabin&,
00101                                 const FareFamilyStruct&);
00102
00106     static void createClass (stdair::FareFamily&,
00107                             const stdair::ClassCode_T&);
00108 };
00109 };
00110
00111 }
00112 #endif // __AIRINV_CMD_INVENTORYGENERATOR_HPP

```

25.117 airinv/command/InventoryManager.cpp File Reference

```
#include <exception>
```



```

#include <algorithm>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/EventQueue.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/BomRootHelper.hpp>
#include <airinv/bom/InventoryHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/command/InventoryManager.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.118 InventoryManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <exception>
00006 #include <algorithm> // To use min
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/basic/BasConst_BomDisplay.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomKeyManager.hpp>
00014 #include <stdair/bom/BomRoot.hpp>
00015 #include <stdair/bom/Inventory.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/SegmentCabin.hpp>
00019 #include <stdair/bom/LegDate.hpp>
00020 #include <stdair/bom/LegCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/BookingClass.hpp>
00023 #include <stdair/bom/GuillotineBlock.hpp>
00024 #include <stdair/bom/TravelSolutionStruct.hpp>
00025 #include <stdair/bom/FareOptionStruct.hpp>
00026 #include <stdair/bom/EventStruct.hpp>
00027 #include <stdair/bom/EventQueue.hpp>
00028 #include <stdair/bom/SnapshotStruct.hpp>

```

```

00029 #include <stdair/bom/RMEventStruct.hpp>
00030 #include <stdair/factory/FacBomManager.hpp>
00031 #include <stdair/factory/FacBom.hpp>
00032 #include <stdair/service/Logger.hpp>
00033 #include <stdair/bom/FareFamily.hpp> // Contains the definition of
    FareFamilyList_T
00034 #include <stdair/bom/BookingClass.hpp> //
00035 // AirInv
00036 #include <airinv/AIRINV_Types.hpp>
00037 #include <airinv/bom/BomRootHelper.hpp>
00038 #include <airinv/bom/InventoryHelper.hpp>
00039 #include <airinv/bom/FlightDateHelper.hpp>
00040 #include <airinv/command/InventoryManager.hpp>
00041
00042 namespace AIRINV {
00043
00044 // //////////////////////////////////////
00045 void InventoryManager::
00046     calculateAvailability (const stdair::BomRoot& iBomRoot,
00047                          stdair::TravelSolutionStruct& ioTravelSolution,
00048                          const stdair::PartnershipTechnique&
00049                          iPartnershipTechnique) {
00050     const stdair::PartnershipTechnique::EN_PartnershipTechnique&
00051     lPartnershipTechnique =
00052         iPartnershipTechnique.getTechnique();
00053
00054     // Browse the list of segments and get the availability for the
00055     // children classes.
00056     const stdair::SegmentPath_T& lSegmentPath =
00057         ioTravelSolution.getSegmentPath();
00058     for (stdair::SegmentPath_T::const_iterator itSK = lSegmentPath.begin();
00059          itSK != lSegmentPath.end(); ++itSK) {
00060         const std::string& lSegmentKey = *itSK;
00061         const stdair::InventoryKey lInvKey =
00062             stdair::BomKeyManager::extractInventoryKey (lSegmentKey);
00063         stdair::Inventory& lInventory =
00064             stdair::BomManager::getObject<stdair::Inventory>(iBomRoot,
00065                                                             lInvKey.toString());
00066
00067         switch (lPartnershipTechnique) {
00068             case stdair::PartnershipTechnique::NONE: {
00069                 InventoryHelper::calculateAvailability (lInventory, lSegmentKey,
00070                                                         ioTravelSolution);
00071                 break;
00072             }
00073             default: {
00074                 InventoryHelper::getYieldAndBidPrice (lInventory, lSegmentKey,
00075                                                         ioTravelSolution);
00076                 break;
00077             }
00078         }
00079     }
00080
00081     switch (lPartnershipTechnique) {
00082     case stdair::PartnershipTechnique::NONE: {
00083         // Compute the availability for each fare option using the AU's.
00084         calculateAvailabilityByAU (ioTravelSolution);
00085         break;
00086     }
00087     case stdair::PartnershipTechnique::RAE_DA:
00088     case stdair::PartnershipTechnique::RAE_YP: {
00089         // 1. Compute the availability for each fare option using RAE
00090         calculateAvailabilityByRAE (ioTravelSolution);
00091         break;
00092     }
00093     case stdair::PartnershipTechnique::IBP_DA:
00094     case stdair::PartnershipTechnique::IBP_YP: {
00095         // 2. Compute the availability for each fare option using protective IBP
00096         calculateAvailabilityByProtectiveIBP (ioTravelSolution);
00097         break;
00098     }
00099     case stdair::PartnershipTechnique::IBP_YP_U:
00100     case stdair::PartnershipTechnique::RMC:
00101     case stdair::PartnershipTechnique::A_RMC: {
00102         // 3. Compute the availability for each fare option using IBP
00103         calculateAvailabilityByIBP (ioTravelSolution);
00104         break;
00105     }
00106     default: {
00107         assert (false);
00108         break;
00109     }
00110 }
00111 }
00112

```

```

00113 // //////////////////////////////////////
00114 void InventoryManager::
00115 calculateAvailabilityByAU (stdair::TravelSolutionStruct& ioTravelSolution) {
00116
00117     // MODIF: segment path string for availability display
00118     std::ostringstream oStr;
00119     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00120     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00121          itSP != lSP.end(); itSP++) {
00122         oStr << *itSP << ";";
00123     }
00124
00125     // Browse the fare options
00126     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef()
;
00127     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00128          itFO != lFOList.end(); ++itFO) {
00129
00130         stdair::FareOptionStruct& lFO = *itFO;
00131
00132         // Check the availability
00133         const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00134
00135         const stdair::ClassAvailabilityMapHolder_T& lClassAvailabilityMapHolder =
00136             ioTravelSolution.getClassAvailabilityMapHolder();
00137
00138         // Initialise the flag stating whether the availability is enough
00139         stdair::Availability_T lAvl =
00140             std::numeric_limits<stdair::Availability_T>::max();
00141
00142         // Sanity check: the travel solution must contain two lists,
00143         // one for the booking class availabilities, the other for the
00144         // fare options.
00145         assert (lClassAvailabilityMapHolder.empty() == false
00146                 && lClassPath.empty() == false);
00147
00148         // List of booking class availability maps (one map per segment)
00149         stdair::ClassAvailabilityMapHolder_T::const_iterator itCAMH =
00150             lClassAvailabilityMapHolder.begin();
00151
00152         // List of fare options
00153         stdair::ClassList_StringList_T::const_iterator itClassList =
00154             lClassPath.begin();
00155
00156         // Browse both lists at the same time, i.e., one element per segment
00157         for (; itCAMH != lClassAvailabilityMapHolder.end()
00158              && itClassList != lClassPath.end(); ++itCAMH, ++itClassList) {
00159
00160             // Retrieve the booking class list for the current segment
00161             const stdair::ClassList_String_T& lCurrentClassList = *itClassList;
00162             assert (lCurrentClassList.size() > 0);
00163
00164             // TODO: instead of just extracting the first booking class,
00165             // perform a choice on the full list of classes.
00166             // Extract one booking class key (class code)
00167             stdair::ClassCode_T lFirstClass;
00168             lFirstClass.append (lCurrentClassList, 0, 1);
00169
00170             // Retrieve the booking class map for the current segment
00171             const stdair::ClassAvailabilityMap_T& lClassAvlMap = *itCAMH;
00172
00173             // Retrieve the availability of the chosen booking class
00174             const stdair::ClassAvailabilityMap_T::const_iterator itClassAvl =
00175                 lClassAvlMap.find (lFirstClass);
00176
00177             if (itClassAvl == lClassAvlMap.end()) {
00178                 // DEBUG
00179                 STDAIR_LOG_DEBUG ("No availability has been set up for the class '"
00180                                 << lFirstClass << "'. Travel solution: "
00181                                 << ioTravelSolution.display());
00182             }
00183             assert (itClassAvl != lClassAvlMap.end());
00184
00185             const stdair::Availability_T& lCurrentAvl = itClassAvl->second;
00186             if (lAvl > lCurrentAvl) {
00187                 lAvl = lCurrentAvl;
00188             }
00189         }
00190
00191         lFO.setAvailability (lAvl);
00192
00193         //MODIF: availability display
00194         STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00195                          << "Availability " << lFO.getAvailability() << ", "
00196                          << "Segment Path " << oStr.str());
00197     }
00198 }

```

```

00199
00200 // \todo: the following code must be either re-written or removed.
00201 //       There is indeed a lot of code duplication.
00202 // //////////////////////////////////////
00203 void InventoryManager::
00204 calculateAvailabilityByRAE (stdair::TravelSolutionStruct& ioTravelSolution) {
00205     std::ostringstream oStr;
00206     const stdair::SegmentPath_T& lsp = ioTravelSolution.getSegmentPath();
00207     for (stdair::SegmentPath_T::const_iterator itSP = lsp.begin();
00208          itSP != lsp.end(); itSP++) {
00209         oStr << *itSP << ",";
00210     }
00211 }
00212
00213 //Retrieve bid price vector and yield maps
00214 const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00215     ioTravelSolution.getClassYieldMapHolder();
00216 const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00217     ioTravelSolution.getClassBpvMapHolder();
00218
00219 //Retrieve the list of fare options and browse it
00220 stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef();
00221
00222 for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00223      itFO != lFOList.end(); ++itFO) {
00224     stdair::FareOptionStruct& lFO = *itFO;
00225
00226     stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00227         lClassYieldMapHolder.begin();
00228     stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00229         lClassBpvMapHolder.begin();
00230
00231     const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00232
00233     // Sanity checks
00234     assert (lClassPath.size() == lClassYieldMapHolder.size());
00235     assert (lClassPath.size() == lClassBpvMapHolder.size());
00236
00237     // Browse class path, class-yield maps, class-(bid price vector) maps.
00238     // Each iteration corresponds to one segment.
00239
00240     std::ostringstream oCPStr;
00241     for (stdair::ClassList_StringList_T::const_iterator itCL =
00242          lClassPath.begin();
00243          itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00244         // Class path determination
00245         if (itCL == lClassPath.begin()) {
00246             oCPStr << *itCL;
00247         } else {
00248             oCPStr << "-" << *itCL;
00249         }
00250         const stdair::ClassList_String_T& lCL = *itCL;
00251         stdair::ClassCode_T lCC;
00252         lCC.append (lCL, 0, 1);
00253         const stdair::ClassYieldMap_T& lCYM = *itCYM;
00254         stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00255         assert (itCCCYM != lCYM.end());
00256         const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00257         stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00258         assert (itCCCBPM != lCBPM.end());
00259         const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00260         assert (lBidPriceVector_ptr != NULL);
00261
00262         // Initialization of fare option availability
00263         if (itCL == lClassPath.begin()) {
00264             lFO.setAvailability (lBidPriceVector_ptr->size());
00265         }
00266
00267         // Availability update
00268         if (lFO.getAvailability() > 0) {
00269             //Segment availability calculation
00270             stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00271             std::reverse_copy (lBidPriceVector_ptr->begin(),
00272                               lBidPriceVector_ptr->end(),
00273                               lReverseBPV.begin());
00274
00275             const stdair::YieldValue_T& lYield = itCCCYM->second;
00276             stdair::BidPriceVector_T::const_iterator lBidPrice =

```

```

00285         std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00286
00287         const stdair::Availability_T lAv1 = lBidPrice - lReverseBPV.begin();
00288
00289         // Availability update
00290         lFO.setAvailability (std::min (lFO.getAvailability(), lAv1));
00291     }
00292 }
00293
00294 // DEBUG
00295 STDAIR_LOG_DEBUG ("Fare option: " << lFO.describe() << ", "
00296                 << "Availability: " << lFO.getAvailability() << ", "
00297                 << "Segment Path: " << oStr.str() << ", ");
00298 }
00299 }
00300
00301 // \todo: the following code must be either re-written or removed.
00302 //       There is indeed a lot of code duplication.
00303 // //////////////////////////////////////
00304 void InventoryManager::
00305 calculateAvailabilityByIBP (stdair::TravelSolutionStruct& ioTravelSolution) {
00306     std::ostringstream oStr;
00307
00308     // Yield valuation coefficient for multi-segment travel solutions
00309     double alpha = 1.0;
00310
00311     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00312     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00313          itSP != lSP.end(); itSP++) {
00314         oStr << *itSP << ";";
00315     }
00316
00317     //Retrieve bid price vector and yield maps
00318     const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00319         ioTravelSolution.getClassYieldMapHolder();
00320     const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00321         ioTravelSolution.getClassBpvMapHolder();
00322
00323     // Retrieve the list of fare options and browse it
00324     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef();
00325
00326     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00327          itFO != lFOList.end(); ++itFO) {
00328         stdair::FareOptionStruct& lFO = *itFO;
00329
00330         stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00331             lClassYieldMapHolder.begin();
00332         stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00333             lClassBpvMapHolder.begin();
00334
00335         const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00336
00337         // Sanity checks
00338         assert (lClassPath.size() == lClassYieldMapHolder.size());
00339         assert (lClassPath.size() == lClassBpvMapHolder.size());
00340
00341         // Yield is taken to be equal to fare (connecting flights)
00342
00343         // \todo: take yield instead
00344         stdair::YieldValue_T lTotalYield = lFO.getFare();
00345         // Bid price initialisation
00346         stdair::BidPrice_T lTotalBidPrice = 0;
00347
00348         // Browse class path, class-yield maps, class-(bid price vector) maps.
00349         // Each iteration corresponds to one segment.
00350
00351         std::ostringstream oCPStr;
00352         for (stdair::ClassList_StringList_T::const_iterator itCL =
00353              lClassPath.begin();
00354              itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00355
00356             // Class path determination
00357             if (itCL == lClassPath.begin()) {
00358                 oCPStr << *itCL;
00359
00360             } else {
00361                 oCPStr << "-" << *itCL;
00362             }
00363
00364             const stdair::ClassList_String_T& lCL = *itCL;
00365             stdair::ClassCode_T lCC;
00366             lCC.append (lCL, 0, 1);
00367
00368             const stdair::ClassYieldMap_T& lCYM = *itCYM;
00369             stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00370             assert (itCCCYM != lCYM.end());

```

```

00371
00372     const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00373     stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00374     assert (itCCCBPM != lCBPM.end());
00375
00376     const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00377     assert (lBidPriceVector_ptr != NULL);
00378
00379     //Initialization of fare option availability
00380     if (itCL == lClassPath.begin()) {
00381         lFO.setAvailability (lBidPriceVector_ptr->size());
00382     }
00383
00384     // Availability update
00385     if (lFO.getAvailability() > 0) {
00386         //Segment availability calculation
00387         stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00388         std::reverse_copy (lBidPriceVector_ptr->begin(),
00389                         lBidPriceVector_ptr->end(), lReverseBPV.begin());
00390
00391         const stdair::YieldValue_T& lYield = itCCCYM->second;
00392         stdair::BidPriceVector_T::const_iterator lBidPrice =
00393             std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00394
00395         const stdair::Availability_T lAvl = lBidPrice - lReverseBPV.begin();
00396
00397         // Availability update
00398         lFO.setAvailability (std::min(lFO.getAvailability(), lAvl));
00399     }
00400
00401     // Total bid price calculation
00402     if (lBidPriceVector_ptr->size() > 0) {
00403         lTotalBidPrice += lBidPriceVector_ptr->back();
00404     }
00405     else {
00406         lTotalBidPrice = std::numeric_limits<stdair::BidPrice_T>::max();
00407     }
00408
00409     // Total yield calculation (has been replaced by total fare).
00410     //lTotalYield += lYield;
00411 }
00412 // Multi-segment bid price control
00413
00414 if (lClassPath.size() > 1) {
00415     if (lFO.getAvailability() > 0) {
00416         const stdair::Availability_T lAvl =
00417             alpha * lTotalYield >= lTotalBidPrice;
00418         lFO.setAvailability (lAvl * lFO.getAvailability());
00419     }
00420     else {
00421         const stdair::Availability_T lAvl =
00422             alpha * lTotalYield >= lTotalBidPrice;
00423         lFO.setAvailability (lAvl);
00424     }
00425
00426     // DEBUG
00427     STDAIR_LOG_DEBUG ("Class: " << oCPStr.str()
00428                     << ", " << "Yield: " << alpha*lTotalYield << ", "
00429                     << "Bid price: " << lTotalBidPrice << ", "
00430                     << "Remaining capacity: " << "Undefined" << " "
00431                     << "Segment date: " << oStr.str());
00432 }
00433
00434 // DEBUG
00435 STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00436                 << "Availability " << lFO.getAvailability() << ", "
00437                 << "Segment Path " << oStr.str() << ", ");
00438 }
00439 }
00440
00441 // \todo: the following code must be either re-written or removed.
00442 //       There is indeed a lot of code duplication.
00443 // //////////////////////////////////////
00444 void InventoryManager::
00445 calculateAvailabilityByProtectiveIBP (stdair::TravelSolutionStruct&
ioTravelSolution) {
00446     std::ostringstream oStr;
00447
00448     // Yield valuation coefficient for multi-segment travel solutions
00449     double alpha = 1.0;
00450
00451     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00452     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00453          itSP != lSP.end(); itSP++) {
00454         oStr << *itSP << ";";
00455     }
00456

```

```

00457 //Retrieve bid price vector and yield maps
00458 const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00459     ioTravelSolution.getClassYieldMapHolder();
00460 const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00461     ioTravelSolution.getClassBpvMapHolder();
00462
00463 //Retrieve the list of fare options and browse it
00464 stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef();
00465 ;
00466 for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00467      itFO != lFOList.end(); ++itFO) {
00468     stdair::FareOptionStruct& lFO = *itFO;
00469
00470     stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00471         lClassYieldMapHolder.begin();
00472     stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00473         lClassBpvMapHolder.begin();
00474
00475     const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00476
00477     // Sanity checks
00478     assert (lClassPath.size() == lClassYieldMapHolder.size());
00479     assert (lClassPath.size() == lClassBpvMapHolder.size());
00480
00481     // Yield is taken to be equal to fare (connecting flights)
00482     // TODO : take yield instead
00483     stdair::YieldValue_T lTotalYield = lFO.getFare();
00484     // Bid price initialisation
00485     stdair::BidPrice_T lTotalBidPrice = 0;
00486     // Maximal bid price initialisation
00487     stdair::BidPrice_T lMaxBidPrice = 0;
00488
00489     //Browse class path, class-yield maps, class-(bid price vector) maps.
00490     //Each iteration corresponds to one segment.
00491
00492     std::ostringstream oCPStr;
00493     for (stdair::ClassList_StringList_T::const_iterator itCL =
00494          lClassPath.begin();
00495          itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00496
00497         // Class path determination
00498         if (itCL == lClassPath.begin()) {
00499             oCPStr << *itCL;
00500
00501         } else {
00502             oCPStr << "-" << *itCL;
00503         }
00504
00505         const stdair::ClassList_String_T& lCL = *itCL;
00506         stdair::ClassCode_T lCC;
00507         lCC.append (lCL, 0, 1);
00508
00509         const stdair::ClassYieldMap_T& lCYM = *itCYM;
00510         stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00511         assert (itCCCYM != lCYM.end());
00512
00513         const stdair::YieldValue_T& lYield = itCCCYM->second;
00514         const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00515         stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00516         assert (itCCCBPM != lCBPM.end());
00517
00518         const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00519         assert (lBidPriceVector_ptr != NULL);
00520
00521         // Initialization of fare option availability
00522         if (itCL == lClassPath.begin()) {
00523             lFO.setAvailability (lBidPriceVector_ptr->size());
00524         }
00525
00526         // Availability update
00527         if (lFO.getAvailability() > 0) {
00528
00529             //Segment availability calculation
00530             stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00531             std::reverse_copy (lBidPriceVector_ptr->begin(),
00532                              lBidPriceVector_ptr->end(), lReverseBPV.begin());
00533
00534             stdair::BidPriceVector_T::const_iterator lBidPrice =
00535                 std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00536
00537             const stdair::Availability_T lAvl = lBidPrice - lReverseBPV.begin();
00538
00539             // Availability update
00540             lFO.setAvailability (std::min(lFO.getAvailability(), lAvl));
00541
00542         }
00543     }

```

```

00543
00544 // Total bid price calculation
00545 if (lBidPriceVector_ptr->size() > 0) {
00546     lTotalBidPrice += lBidPriceVector_ptr->back();
00547
00548     if (lMaxBidPrice < lBidPriceVector_ptr->back()) {
00549         lMaxBidPrice = lBidPriceVector_ptr->back();
00550     }
00551
00552 } else {
00553     lTotalBidPrice = std::numeric_limits<stdair::BidPrice_T>::max();
00554 }
00555
00556 // Total yield calculation (has been replaced by total fare).
00557 //lTotalYield += lYield;
00558 }
00559 // Multi-segment bid price control
00560
00561 // Protective IBP (maximin): guarantees the minimal yield for each
airline
00562 // Proration factors are all equal to 1/{number of partners}.
00563
00564 lTotalBidPrice = std::max (lMaxBidPrice * lClassPath.size(),
00565                             lTotalBidPrice);
00566
00567 if (lClassPath.size() > 1) {
00568     if (lFO.getAvailability() > 0) {
00569         const stdair::Availability_T lAvl =
00570             alpha * lTotalYield >= lTotalBidPrice;
00571         lFO.setAvailability (lAvl * lFO.getAvailability());
00572
00573     } else {
00574         const stdair::Availability_T lAvl =
00575             alpha * lTotalYield >= lTotalBidPrice;
00576         lFO.setAvailability (lAvl);
00577     }
00578
00579     // DEBUG
00580     STDAIR_LOG_DEBUG ("Class: " << oCPStr.str()
00581                       << ", " << "Yield: " << alpha*lTotalYield << ", "
00582                       << "Bid price: " << lTotalBidPrice << ", "
00583                       << "Remaining capacity: " << "Undefined" << " "
00584                       << "Segment date: " << oStr.str());
00585 }
00586
00587 // DEBUG
00588 STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00589                  << "Availability " << lFO.getAvailability() << ", "
00590                  << "Segment Path " << oStr.str() << ", ");
00591 }
00592 }
00593
00594 //MODIF
00595 // //////////////////////////////////////
00596 void InventoryManager::setDefaultBidPriceVector (stdair::BomRoot& ioBomRoot)
{
00597
00598     const stdair::InventoryList_T& lInvList =
00599         stdair::BomManager::getList<stdair::Inventory> (ioBomRoot);
00600     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00601          itInv != lInvList.end(); ++itInv) {
00602         stdair::Inventory* lCurrentInv_ptr = *itInv;
00603         assert (lCurrentInv_ptr != NULL);
00604
00605         // Set the default bid price for own cabins.
00606         setDefaultBidPriceVector (*lCurrentInv_ptr);
00607
00608         // Check if the inventory contains images of partner inventories.
00609         // If so, set the default bid price for their cabins.
00610         if (stdair::BomManager::hasList<stdair::Inventory> (*lCurrentInv_ptr)) {
00611             const stdair::InventoryList_T& lPartnerInvList =
00612                 stdair::BomManager::getList<stdair::Inventory> (*lCurrentInv_ptr);
00613
00614             for (stdair::InventoryList_T::const_iterator itPartnerInv =
00615                  lPartnerInvList.begin();
00616                  itPartnerInv != lPartnerInvList.end(); ++itPartnerInv) {
00617                 stdair::Inventory* lCurrentPartnerInv_ptr = *itPartnerInv;
00618                 assert (lCurrentPartnerInv_ptr != NULL);
00619
00620                 setDefaultBidPriceVector (*lCurrentPartnerInv_ptr);
00621             }
00622         }
00623     }
00624 }
00625
00626 // //////////////////////////////////////
00627 void InventoryManager::

```



```

00628     setDefaultBidPriceVector (stdair::Inventory& ioInventory) {
00629
00630         const stdair::FlightDateList_T& lFlightDateList =
00631             stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00632         for (stdair::FlightDateList_T::const_iterator itFlightDate =
00633             lFlightDateList.begin();
00634             itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00635             stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00636             assert (lCurrentFlightDate_ptr != NULL);
00637
00638             // Check if the flight date holds a list of leg dates.
00639             // If so retrieve it and initialise the bid price vectors of their
cabins.
00640             if (stdair::BomManager::hasList<stdair::LegDate> (*lCurrentFlightDate_ptr
)) {
00641                 const stdair::LegDateList_T& lLegDateList =
00642                     stdair::BomManager::getList<stdair::LegDate>
(*lCurrentFlightDate_ptr);
00643                 for (stdair::LegDateList_T::const_iterator itLegDate =
00644                     lLegDateList.begin();
00645                     itLegDate != lLegDateList.end(); ++itLegDate) {
00646                     stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00647                     assert (lCurrentLegDate_ptr != NULL);
00648
00649                     const stdair::LegCabinList_T& lLegCabinList =
00650                         stdair::BomManager::getList<stdair::LegCabin>
(*lCurrentLegDate_ptr);
00651                     for (stdair::LegCabinList_T::const_iterator itLegCabin =
00652                         lLegCabinList.begin();
00653                         itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00654                         stdair::LegCabin* lCurrentLegCabin_ptr = *itLegCabin;
00655                         assert (lCurrentLegCabin_ptr != NULL);
00656
00657                         const stdair::CabinCapacity_T& lCabinCapacity =
00658                             lCurrentLegCabin_ptr->getPhysicalCapacity();
00659                         lCurrentLegCabin_ptr->emptyBidPriceVector();
00660
00661                         stdair::BidPriceVector_T& lBPV =
00662                             lCurrentLegCabin_ptr->getBidPriceVector();
00663
00664                         //for (stdair::CabinCapacity_T k = 0; k!=lCabinCapacity; k++)
{ lBPV.push_back(400 + 300/sqrt(k+1)); }
00665                         for (stdair::CabinCapacity_T k = 0; k != lCabinCapacity; k++) {
00666                             lBPV.push_back (400);
00667                         }
00668
00669                         lCurrentLegCabin_ptr->setPreviousBidPrice (lBPV.back());
00670                         lCurrentLegCabin_ptr->setCurrentBidPrice (lBPV.back());
00671                     }
00672                 }
00673             }
00674         }
00675     }
00676
00677     // //////////////////////////////////////
00678     bool InventoryManager::sell (stdair::Inventory& ioInventory,
00679                                 const std::string& iSegmentDateKey,
00680                                 const stdair::ClassCode_T& iClassCode,
00681                                 const stdair::PartySize_T& iPartySize) {
00682
00683         // Make the sale within the inventory.
00684         return InventoryHelper::sell (ioInventory, iSegmentDateKey,
00685                                     iClassCode, iPartySize);
00686     }
00687
00688     // //////////////////////////////////////
00689     bool InventoryManager::cancel (stdair::Inventory& ioInventory,
00690                                   const std::string& iSegmentDateKey,
00691                                   const stdair::ClassCode_T& iClassCode,
00692                                   const stdair::PartySize_T& iPartySize) {
00693
00694         // Make the sale within the inventory.
00695         return InventoryHelper::cancel (ioInventory, iSegmentDateKey,
00696                                         iClassCode, iPartySize);
00697     }
00698
00699     // //////////////////////////////////////
00700     void InventoryManager::
00701     updateBookingControls (stdair::FlightDate& ioFlightDate) {
00702
00703         // Forward the call to FlightDateHelper.
00704         FlightDateHelper::updateBookingControls (ioFlightDate);
00705     }
00706
00707     // //////////////////////////////////////
00708     void InventoryManager::takeSnapshots(const stdair::Inventory& iInventory,
00709                                         const stdair::DateTime_T& iSnapshotTime)

```

```

{
00710
00711     // Make the snapshots within the inventory
00712     InventoryHelper::takeSnapshots (iInventory, iSnapshotTime);
00713 }
00714
00715 // //////////////////////////////////////
00716 void InventoryManager::
00717 createDirectAccesses (const stdair::BomRoot& iBomRoot) {
00718
00719     // Browse the list of inventories and create direct accesses
00720     // within each inventory.
00721     const stdair::InventoryList_T& lInvList =
00722         stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00723     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00724          itInv != lInvList.end(); ++itInv) {
00725         stdair::Inventory* lCurrentInv_ptr = *itInv;
00726         assert (lCurrentInv_ptr != NULL);
00727
00728         createDirectAccesses (*lCurrentInv_ptr);
00729     }
00730
00731     // Fill some attributes of segment-date with the routing legs.
00732     BomRootHelper::fillFromRouting (iBomRoot);
00733 }
00734
00735 // //////////////////////////////////////
00736 void InventoryManager::
00737 createDirectAccesses (stdair::Inventory& ioInventory) {
00738
00739     // Browse the list of flight-dates and create direct accesses
00740     // within each flight-date.
00741     const stdair::FlightDateList_T& lFlightDateList =
00742         stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00743     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00744          lFlightDateList.begin();
00745          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00746         stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00747         assert (lCurrentFlightDate_ptr != NULL);
00748
00749         createDirectAccesses (*lCurrentFlightDate_ptr);
00750     }
00751 }
00752
00753 // //////////////////////////////////////
00754 void InventoryManager::
00755 createDirectAccesses (stdair::FlightDate& ioFlightDate) {
00756
00757     // Browse the list of segment-dates and create direct accesses
00758     // within each segment-date.
00759     const stdair::SegmentDateList_T& lSegmentDateList =
00760         stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00761     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00762          lSegmentDateList.begin();
00763          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00764
00765         stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00766         assert (lCurrentSegmentDate_ptr != NULL);
00767
00768         /*
00769          * If the segment is just marketed by this carrier,
00770          * retrieve the operating segment and call the createDirectAcces
00771          * method on its parent (flight date).
00772          */
00773         const stdair::SegmentDate* lOperatingSegmentDate_ptr =
00774             lCurrentSegmentDate_ptr->getOperatingSegmentDate ();
00775         if (lOperatingSegmentDate_ptr != NULL) {
00776             // Then get the (parent) flight date and create direct access.
00777             stdair::FlightDate* lOperatingFlightDate_ptr =
00778                 stdair::BomManager::getParentPtr<stdair::FlightDate>
00779                 (*lOperatingSegmentDate_ptr);
00780             assert (lOperatingFlightDate_ptr != NULL);
00781             createDirectAccesses (*lOperatingFlightDate_ptr);
00782         } else {
00783
00784             const stdair::AirportCode_T& lBoardingPoint =
00785                 lCurrentSegmentDate_ptr->getBoardingPoint ();
00786
00787             stdair::AirportCode_T currentBoardingPoint = lBoardingPoint;
00788             const stdair::AirportCode_T& lOffPoint =
00789                 lCurrentSegmentDate_ptr->getOffPoint ();
00790
00791             // Add a sanity check so as to ensure that the loop stops. If
00792             // there are more than MAXIMAL_NUMBER_OF_LEGS legs, there is
00793             // an issue somewhere in the code (not in the parser, as the
00794             // segments are derived from the legs thanks to the
00795             // FlightPeriodStruct::buildSegments() method).

```

```

00795         unsigned short i = 1;
00796         while (currentBoardingPoint != lOffPoint
00797             && i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT) {
00798             // Retrieve the (unique) LegDate getting that Boarding Point
00799             stdair::LegDate& lLegDate = stdair::BomManager::
00800                 getObject<stdair::LegDate> (ioFlightDate, currentBoardingPoint);
00801
00802             // Link the SegmentDate and LegDate together
00803             stdair::FacBomManager::addToListAndMap (*lCurrentSegmentDate_ptr,
00804                 lLegDate);
00805             stdair::FacBomManager::addToListAndMap (lLegDate,
00806                 *lCurrentSegmentDate_ptr);
00807
00808             // Prepare the next iteration
00809             currentBoardingPoint = lLegDate.getOffPoint();
00810             ++i;
00811         }
00812         assert (i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT);
00813
00814         // Create the routing for the leg- and segment-cabins.
00815         // At the same time, set the SegmentDate attributes derived from
00816         // its routing legs (e.g., boarding and off dates).
00817         createDirectAccesses (*lCurrentSegmentDate_ptr);
00818     }
00819 }
00820 }
00821
00822 // //////////////////////////////////////
00823 void InventoryManager::
00824 createDirectAccesses (stdair::SegmentDate& ioSegmentDate) {
00825
00826     // Browse the list of segment-cabins and create direct accesses
00827     // within each segment-cabin.
00828     const stdair::SegmentCabinList_T& lSegmentCabinList =
00829         stdair::BomManager::getList<stdair::SegmentCabin> (ioSegmentDate);
00830     for (stdair::SegmentCabinList_T::const_iterator itSegmentCabin =
00831         lSegmentCabinList.begin();
00832         itSegmentCabin != lSegmentCabinList.end(); ++itSegmentCabin) {
00833
00834         //
00835         stdair::SegmentCabin* lCurrentSegmentCabin_ptr = *itSegmentCabin;
00836         assert (lCurrentSegmentCabin_ptr != NULL);
00837
00838         //
00839         const stdair::CabinCode_T& lCabinCode =
00840             lCurrentSegmentCabin_ptr->getCabinCode();
00841
00842         // Iterate on the routing legs
00843         const stdair::LegDateList_T& lLegDateList =
00844             stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00845         for (stdair::LegDateList_T::const_iterator itLegDate =
00846             lLegDateList.begin();
00847             itLegDate != lLegDateList.end(); ++itLegDate) {
00848
00849             const stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00850             assert (lCurrentLegDate_ptr != NULL);
00851
00852             // Retrieve the LegCabin getting the same class of service
00853             // (cabin code) as the SegmentCabin.
00854             stdair::LegCabin& lLegCabin = stdair::BomManager::
00855                 getObject<stdair::LegCabin> (*lCurrentLegDate_ptr, lCabinCode);
00856
00857             stdair::FacBomManager::addToListAndMap (*lCurrentSegmentCabin_ptr,
00858                 lLegCabin,
00859                 lLegCabin.getFullerKey());
00860
00861             stdair::FacBomManager::
00862                 addToListAndMap (lLegCabin, *lCurrentSegmentCabin_ptr,
00863                     lCurrentSegmentCabin_ptr->getFullerKey());
00864         }
00865     }
00866 }
00867
00868 // //////////////////////////////////////
00869 void InventoryManager::
00870 buildSimilarSegmentCabinSets (const stdair::BomRoot& iBomRoot) {
00871     // Browse the list of inventories and create direct accesses
00872     // within each inventory.
00873     const stdair::InventoryList_T& lInvList =
00874         stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00875     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00876         itInv != lInvList.end(); ++itInv) {
00877         stdair::Inventory* lCurrentInv_ptr = *itInv;
00878         assert (lCurrentInv_ptr != NULL);
00879
00880         buildSimilarSegmentCabinSets (*lCurrentInv_ptr);
00881     }
00882 }
00883
00884 // //////////////////////////////////////
00885 void InventoryManager::
00886 buildSimilarSegmentCabinSets (*lCurrentInv_ptr);
00887 }
00888
00889 // //////////////////////////////////////
00890 void InventoryManager::
00891 buildSimilarSegmentCabinSets (*lCurrentInv_ptr);
00892 }
00893
00894 // //////////////////////////////////////
00895 void InventoryManager::
00896 buildSimilarSegmentCabinSets (*lCurrentInv_ptr);
00897 }
00898
00899 // //////////////////////////////////////
00900 void InventoryManager::
00901 buildSimilarSegmentCabinSets (*lCurrentInv_ptr);
00902 }

```

```

00902     }
00903
00904     // //////////////////////////////////////
00905     void InventoryManager::
00906     buildSimilarSegmentCabinSets (stdair::Inventory& ioInventory) {
00907         // For instance, we consider two flight-dates are
00908         // similar if they have the same flight number and the same
00909         // day-of-the-week departure.
00910
00911         // Browse the segment-cabin list and build the sets of segment-cabin
00912         // which have the same flight number and origin-destination
00913         SimilarSegmentCabinSetMap_T lSSCSM;
00914
00915         // Browsing the flight-date list
00916         const stdair::FlightDateList_T& lFlightDateList =
00917             stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00918         for (stdair::FlightDateList_T::const_iterator itFD= lFlightDateList.begin()
00919 ;
00920             itFD != lFlightDateList.end(); ++itFD) {
00921             const stdair::FlightDate* lFD_ptr = *itFD;
00922             assert (lFD_ptr != NULL);
00923             const stdair::FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
00924
00925             // Browsing the segment-date list and retrieve the departure
00926             // date, the origine and the destination of the segment
00927             const stdair::SegmentDateList_T& lSegmentDateList =
00928                 stdair::BomManager::getList<stdair::SegmentDate> (*lFD_ptr);
00929             for (stdair::SegmentDateList_T::const_iterator itSD =
00930                 lSegmentDateList.begin(); itSD != lSegmentDateList.end(); ++itSD)
00931 {
00932             const stdair::SegmentDate* lSD_ptr = *itSD;
00933             assert (lSD_ptr != NULL);
00934
00935             const stdair::Date_T& lDepartureDate = lSD_ptr->getBoardingDate();
00936             const stdair::AirportCode_T& lOrigin = lSD_ptr->getBoardingPoint();
00937             const stdair::AirportCode_T& lDestination = lSD_ptr->getOffPoint();
00938
00939             // Browsing the segment-cabin list and retrieve the cabin code and
00940             // build the corresponding key map.
00941             const stdair::SegmentCabinList_T& lSegmentCabinList =
00942                 stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
00943             for (stdair::SegmentCabinList_T::const_iterator itSC =
00944                 lSegmentCabinList.begin();
00945                 itSC != lSegmentCabinList.end(); ++itSC) {
00946                 stdair::SegmentCabin* lSC_ptr = *itSC;
00947                 assert (lSC_ptr != NULL);
00948
00949                 std::ostringstream oStr;
00950                 oStr << lFlightNumber << lDepartureDate.day_of_week()
00951                     << lOrigin << lDestination << lSC_ptr->getCabinCode();
00952                 const std::string lMapKey = oStr.str();
00953
00954                 // Add the segment cabin to the similar segment cabin set map.
00955                 SimilarSegmentCabinSetMap_T::iterator itSSCS = lSSCSM.find (lMapKey);
00956                 if (itSSCS == lSSCSM.end()) {
00957                     DepartureDateSegmentCabinMap_T lDDSCMap;
00958                     lDDSCMap.insert (DepartureDateSegmentCabinMap_T::
00959                         value_type (lDepartureDate, lSC_ptr));
00960                     lSSCSM.insert (SimilarSegmentCabinSetMap_T::
00961                         value_type (lMapKey, lDDSCMap));
00962                 } else {
00963                     DepartureDateSegmentCabinMap_T& lDDSCMap = itSSCS->second;
00964                     lDDSCMap.insert (DepartureDateSegmentCabinMap_T::
00965                         value_type (lDepartureDate, lSC_ptr));
00966                 }
00967             }
00968         }
00969
00970         // Initialise the guillotine blocks.
00971         stdair::GuillotineNumber_T lGuillotineNumber = 1;
00972         for (SimilarSegmentCabinSetMap_T::const_iterator itSSCS = lSSCSM.begin();
00973             itSSCS != lSSCSM.end(); ++itSSCS, ++lGuillotineNumber) {
00974             const DepartureDateSegmentCabinMap_T& lDDSCMap = itSSCS->second;
00975             buildGuillotineBlock (ioInventory, lGuillotineNumber, lDDSCMap);
00976         }
00977     }
00978
00979     // //////////////////////////////////////
00980     void InventoryManager::
00981     buildGuillotineBlock (stdair::Inventory& ioInventory,
00982         const stdair::GuillotineNumber_T& iGuillotineNumber,
00983         const DepartureDateSegmentCabinMap_T& iDDSCMap) {
00984         // Build an empty guillotine block.
00985         const stdair::GuillotineBlockKey lKey (iGuillotineNumber);
00986         stdair::GuillotineBlock& lGuillotineBlock =

```

```

00987     stdair::FacBom<stdair::GuillotineBlock>::instance().create (lKey);
00988     stdair::FacBomManager::addToListAndMap (ioInventory, lGuillotineBlock);
00989
00990     // Build the value type index map.
00991     DepartureDateSegmentCabinMap_T::const_iterator itDDSC = iDDSCMap.begin();
00992     assert (itDDSC != iDDSCMap.end());
00993     const stdair::SegmentCabin* lSegmentCabin_ptr = itDDSC->second;
00994
00995     // Browse the booking class list and build the value type for the classes
00996     // as well as for the cabin (Q-equivalent).
00997     stdair::ValueTypeIndexMap_T lValueTypeIndexMap;
00998     stdair::BlockIndex_T lBlockIndex = 0;
00999     std::ostringstream lSCMapKey;
01000     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
01001         << lSegmentCabin_ptr->describeKey();
01002     lValueTypeIndexMap.insert (stdair::ValueTypeIndexMap_T::
01003         value_type (lSCMapKey.str(), lBlockIndex));
01004     ++lBlockIndex;
01005
01006     // Browse the booking class list
01007     const stdair::BookingClassList_T& lBCList =
01008         stdair::BomManager::getList<stdair::BookingClass>(*lSegmentCabin_ptr);
01009     for (stdair::BookingClassList_T::const_iterator itBC= lBCList.begin();
01010         itBC != lBCList.end(); ++itBC) {
01011         const stdair::BookingClass* lBookingClass_ptr = *itBC;
01012         assert (lBookingClass_ptr != NULL);
01013         lValueTypeIndexMap.
01014             insert (stdair::ValueTypeIndexMap_T::
01015                 value_type (lBookingClass_ptr->describeKey(), lBlockIndex));
01016         ++lBlockIndex;
01017     }
01018
01019     // Build the segment-cabin index map
01020     stdair::SegmentCabinIndexMap_T lSegmentCabinIndexMap;
01021     stdair::BlockNumber_T lBlockNumber = 0;
01022     for (; itDDSC != iDDSCMap.end(); ++itDDSC, ++lBlockNumber) {
01023         stdair::SegmentCabin* lCurrentSC_ptr = itDDSC->second;
01024         assert (lCurrentSC_ptr != NULL);
01025         lSegmentCabinIndexMap.
01026             insert (stdair::SegmentCabinIndexMap_T::value_type (lCurrentSC_ptr,
01027                 lBlockNumber));
01028
01029         // Added the guillotine to the segment-cabin.
01030         lCurrentSC_ptr->setGuillotineBlock (lGuillotineBlock);
01031     }
01032
01033     // Initialise the guillotine block.
01034     lGuillotineBlock.initSnapshotBlocks (lSegmentCabinIndexMap,
01035         lValueTypeIndexMap);
01036 }
01037
01038 // //////////////////////////////////////
01039 void InventoryManager::initSnapshotEvents (const stdair::Date_T& iStartDate,
01040     const stdair::Date_T& iEndDate,
01041     stdair::EventQueue& ioQueue) {
01042     const stdair::Duration_T lTimeZero (0, 0, 0);
01043     const stdair::Duration_T lOneDayDuration (24, 0, 0);
01044     const stdair::DateTime_T lBeginSnapshotTime (iStartDate, lTimeZero);
01045     const stdair::DateTime_T lEndSnapshotTime (iEndDate, lTimeZero);
01046
01047     // TODO: remove the default airline code.
01048     stdair::NbOfEvents_T lNbOfSnapshots = 0.0;
01049     for (stdair::DateTime_T lSnapshotTime = lBeginSnapshotTime;
01050         lSnapshotTime < lEndSnapshotTime; lSnapshotTime += lOneDayDuration) {
01051         // Create the snapshot event structure
01052         stdair::SnapshotPtr_T lSnapshotStruct =
01053             boost::make_shared<stdair::SnapshotStruct>(stdair::DEFAULT_AIRLINE_CODE
01054                 ,
01055                 lSnapshotTime);
01056         // Create the event structure
01057         stdair::EventStruct lEventStruct (stdair::EventType::SNAPSHOT,
01058             lSnapshotStruct);
01059
01060         ioQueue.addEvent (lEventStruct);
01061         ++lNbOfSnapshots;
01062     }
01063
01064     ioQueue.addStatus (stdair::EventType::SNAPSHOT, lNbOfSnapshots);
01065 }
01066
01067 // //////////////////////////////////////
01068 void InventoryManager::
01069     initRMEvents (const stdair::Inventory& iInventory,
01070         stdair::RMEventList_T& ioRMEventList,
01071         const stdair::Date_T& iStartDate,
01072         const stdair::Date_T& iEndDate) {
01073     const stdair::Duration_T lTimeZero (0, 0, 0);

```

```

01080     const stdair::Duration_T lTime (0, 0, 10);
01081     const stdair::Duration_T lOneDayDuration (24, 0, 0);
01082     const stdair::DateTime_T lEarliestEventTime (iStartDate, lTimeZero);
01083     const stdair::DateTime_T lLatestEventTime (iEndDate, lTimeZero);
01084
01085     const stdair::AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
01086
01087     // Browse the list of flight-dates and initialise the RM events for
01088     // each flight-date.
01089     const stdair::FlightDateList_T& lFDList =
01090         stdair::BomManager::getList<stdair::FlightDate> (iInventory);
01091     for (stdair::FlightDateList_T::const_iterator itFD = lFDList.begin();
01092          itFD != lFDList.end(); ++itFD) {
01093         const stdair::FlightDate* lFD_ptr = *itFD;
01094         assert (lFD_ptr != NULL);
01095
01096         // Retrieve the departure date and initialise the RM events with
01097         // the data collection points of the inventory.
01098         const stdair::Date_T& lDepartureDate = lFD_ptr->getDepartureDate();
01099         const stdair::DateTime_T lDepartureDateTime (lDepartureDate, lTime);
01100         for (stdair::DCPList_T::const_iterator itDCP =
01101              stdair::DEFAULT_DCP_LIST.begin();
01102              itDCP != stdair::DEFAULT_DCP_LIST.end(); ++itDCP) {
01103             const stdair::DCP_T& lDCP = *itDCP;
01104
01105             // Create the event time and check if it is in the validate interval
01106             const stdair::DateTime_T lEventTime =
01107                 lDepartureDateTime - lOneDayDuration * lDCP;
01108             if (lEventTime >= lEarliestEventTime && lEventTime <= lLatestEventTime)
01109             {
01110                 const stdair::KeyDescription_T lKeyDes = lFD_ptr->describeKey();
01111                 stdair::RMEventStruct lRMEvent (lAirlineCode, lKeyDes, lEventTime);
01112                 ioRMEventList.push_back (lRMEvent);
01113             }
01114         }
01115     }
01116
01117     // //////////////////////////////////////
01118     void InventoryManager::
01119     addRMEventsToEventQueue (stdair::EventQueue& ioQueue,
01120                             stdair::RMEventList_T& ioRMEventList) {
01121         // Browse the RM event list and add them to the queue.
01122         for (stdair::RMEventList_T::iterator itRMEvent = ioRMEventList.begin();
01123              itRMEvent != ioRMEventList.end(); ++itRMEvent) {
01124             stdair::RMEventStruct& lRMEvent = *itRMEvent;
01125             stdair::RMEventPtr_T lRMEventPtr =
01126                 boost::make_shared<stdair::RMEventStruct> (lRMEvent);
01127             stdair::EventStruct lEventStruct (stdair::EventType::RM, lRMEventPtr);
01128             ioQueue.addEvent (lEventStruct);
01129         }
01130
01131         // Update the status of RM events within the event queue.
01132         ioQueue.updateStatus (stdair::EventType::RM, ioRMEventList.size());
01133     }
01134 }

```

25.119 airinv/command/InventoryManager.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>

```

Classes

- class [AIRINV::InventoryManager](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

Typedefs

- typedef std::map< const
stdair::Date_T,
stdair::SegmentCabin * > AIRINV::DepartureDateSegmentCabinMap_T
- typedef std::map< const
std::string,
DepartureDateSegmentCabinMap_T > AIRINV::SimilarSegmentCabinSetMap_T

25.120 InventoryManager.hpp

```

00001 #ifndef __AIRINV_CMD_INVENTORYMANAGER_HPP
00002 #define __AIRINV_CMD_INVENTORYMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // STDAIR
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/bom/RMEventTypes.hpp>
00012 #include <stdair/basic/PartnershipTechnique.hpp>
00013
00014 // Forward declarations
00015 namespace stdair {
00016     class BomRoot;
00017     class Inventory;
00018     class FlightDate;
00019     class SegmentDate;
00020     class SegmentCabin;
00021     class EventQueue;
00022     struct TravelSolutionStruct;
00023 }
00024
00025 namespace AIRINV {
00026
00027     // ////////////////////////////////// Type definitions //////////////////////////////////
00028     typedef std::map<const stdair::Date_T,
00029                     stdair::SegmentCabin*> DepartureDateSegmentCabinMap_T;
00030     typedef std::map<const std::string,
00031                     DepartureDateSegmentCabinMap_T> SimilarSegmentCabinSetMap_T;
00032
00033     class InventoryManager {
00034     friend class AIRINV_Master_Service;
00035     friend class AIRINV_Service;
00036
00037     private:
00038         static void initSnapshotEvents (const stdair::Date_T&,
00039                                         const stdair::Date_T&,
00040                                         stdair::EventQueue&);
00041
00042         static void initRMEvents (const stdair::Inventory&, stdair::RMEventList_T&,
00043                                   const stdair::Date_T&, const stdair::Date_T&);
00044
00045         static void addRMEventsToEventQueue (stdair::EventQueue&,
00046                                               stdair::RMEventList_T&);
00047
00048         static void calculateAvailability (const stdair::BomRoot&,
00049                                           stdair::TravelSolutionStruct&,
00050                                           const stdair::PartnershipTechnique&);
00051
00052         static void calculateAvailabilityByAU (stdair::TravelSolutionStruct&);
00053
00054         static void calculateAvailabilityByRAE (stdair::TravelSolutionStruct&);
00055
00056         static void calculateAvailabilityByIBP (stdair::TravelSolutionStruct&);
00057
00058         static void calculateAvailabilityByProtectiveIBP (
00059             stdair::TravelSolutionStruct&);
00060
00061         static bool sell (stdair::Inventory&, const std::string& iSegmentDateKey,
00062                          const stdair::ClassCode_T&, const stdair::PartySize_T&);
00063
00064         static bool cancel (stdair::Inventory&, const std::string& iSegmentDateKey,
00065                             const stdair::ClassCode_T&, const stdair::PartySize_T&);
00066
00067     };
00068
00069     static void takeSnapshots (const stdair::Inventory&,
00070                                const stdair::DateTime_T&);
00071
00072

```

```

00090     static void updateBookingControls (stdair::FlightDate&);
00091
00092 public:
00093     static void createDirectAccesses (const stdair::BomRoot&);
00094     static void createDirectAccesses (stdair::Inventory&);
00095     static void createDirectAccesses (stdair::FlightDate&);
00096     static void createDirectAccesses (stdair::SegmentDate&);
00097
00098     static void buildSimilarSegmentCabinSets (const stdair::BomRoot&);
00099     static void buildSimilarSegmentCabinSets (stdair::Inventory&);
00100     static void buildGuillotineBlock (stdair::Inventory&,
00101                                     const stdair::GuillotineNumber_T&,
00102                                     const DepartureDateSegmentCabinMap_T&);
00103
00104     static void setDefaultBidPriceVector (stdair::BomRoot&);
00105     static void setDefaultBidPriceVector (stdair::Inventory&);
00106
00107 private:
00108     InventoryManager() {}
00109     InventoryManager(const InventoryManager&) {}
00110     ~InventoryManager() {}
00111 };
00112
00113 #endif // __AIRINV_CMD_INVENTORYMANAGER_HPP

```

25.121 airinv/command/InventoryParser.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/InventoryParserHelper.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.122 InventoryParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/InventoryParserHelper.hpp>
00013 #include <airinv/command/InventoryParser.hpp>
00014 #include <airinv/command/InventoryManager.hpp>
00015
00016 namespace AIRINV {
00017
00018 // //////////////////////////////////////
00019 void InventoryParser::
00020 buildInventory (const stdair::Filename_T& iInventoryFilename,
00021               stdair::BomRoot& ioBomRoot) {
00022
00023     // Check that the file path given as input corresponds to an actual file
00024     const bool doesExistAndIsReadable =
00025         stdair::BasFileMgr::doesExistAndIsReadable (iInventoryFilename);
00026     if (doesExistAndIsReadable == false) {
00027         std::ostringstream oMessage;
00028         oMessage << "The inventory input file, '" << iInventoryFilename

```



```

00029         << "", can not be retrieved on the file-system";
00030         STDAIR_LOG_ERROR (oMessage.str());
00031         throw InventoryInputFileNotFoundException (oMessage.str());
00032     }
00033
00034     // Initialise the inventory file parser.
00035     InventoryFileParser lInventoryParser (ioBomRoot, iInventoryFilename);
00036
00037     // Parse the CSV-formatted inventory input file, and generate the
00038     // corresponding Inventory-related objects.
00039     lInventoryParser.buildInventory();
00040
00041     // Complete the BomRoot BOM building: create the routings for all
00042     // the inventories.
00043     InventoryManager::createDirectAccesses (ioBomRoot);
00044 }
00045
00046 }

```

25.123 airinv/command/InventoryParser.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [AIRINV::InventoryParser](#)
Class wrapping the parser entry point.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.124 InventoryParser.hpp

```

00001 #ifndef __AIRINV_CMD_INVENTORYPARSER_HPP
00002 #define __AIRINV_CMD_INVENTORYPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00012 namespace stdair {
00013     class BomRoot;
00014 }
00015
00016 namespace AIRINV {
00017
00021     class InventoryParser : public stdair::CmdAbstract {
00022     public:
00032         static void buildInventory (const stdair::Filename_T& iInventoryFilename,
00033                                     stdair::BomRoot&);
00034     };
00035 }
00036 #endif // __AIRINV_CMD_INVENTORYPARSER_HPP

```

25.125 airinv/command/InventoryParserHelper.cpp File Reference

```

#include <cassert>

```

```
#include <stdair/service/Logger.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <airinv/command/InventoryBuilder.hpp>
#include <airinv/command/InventoryParserHelper.hpp>
```

Namespaces

- namespace [AIRINV](#)
- namespace [AIRINV::InventoryParserHelper](#)

Functions

- repeat_p_t [AIRINV::InventoryParserHelper::airline_code_p](#) (chset_t("0-9A-Z").derived(), 2, 3)
- bounded1_4_p_t [AIRINV::InventoryParserHelper::flight_number_p](#) (uint1_4_p.derived(), 0u, 9999u)
- bounded2_p_t [AIRINV::InventoryParserHelper::year_p](#) (uint2_p.derived(), 0u, 99u)
- bounded2_p_t [AIRINV::InventoryParserHelper::month_p](#) (uint2_p.derived(), 1u, 12u)
- bounded2_p_t [AIRINV::InventoryParserHelper::day_p](#) (uint2_p.derived(), 1u, 31u)
- repeat_p_t [AIRINV::InventoryParserHelper::dow_p](#) (chset_t("0-1").derived().derived(), 7, 7)
- repeat_p_t [AIRINV::InventoryParserHelper::airport_p](#) (chset_t("0-9A-Z").derived(), 3, 3)
- bounded1_2_p_t [AIRINV::InventoryParserHelper::hours_p](#) (uint1_2_p.derived(), 0u, 24u)
- bounded2_p_t [AIRINV::InventoryParserHelper::minutes_p](#) (uint2_p.derived(), 0u, 59u)
- bounded2_p_t [AIRINV::InventoryParserHelper::seconds_p](#) (uint2_p.derived(), 0u, 59u)
- chset_t [AIRINV::InventoryParserHelper::cabin_code_p](#) ("A-Z")
- chset_t [AIRINV::InventoryParserHelper::class_code_p](#) ("A-Z")
- chset_t [AIRINV::InventoryParserHelper::passenger_type_p](#) ("A-Z")
- repeat_p_t [AIRINV::InventoryParserHelper::class_code_list_p](#) (chset_t("A-Z").derived(), 1, 26)
- bounded1_3_p_t [AIRINV::InventoryParserHelper::stay_duration_p](#) (uint1_3_p.derived(), 0u, 999u)

Variables

- int1_p_t [AIRINV::InventoryParserHelper::int1_p](#)
- uint2_p_t [AIRINV::InventoryParserHelper::uint2_p](#)
- uint1_2_p_t [AIRINV::InventoryParserHelper::uint1_2_p](#)
- uint1_3_p_t [AIRINV::InventoryParserHelper::uint1_3_p](#)
- uint4_p_t [AIRINV::InventoryParserHelper::uint4_p](#)
- uint1_4_p_t [AIRINV::InventoryParserHelper::uint1_4_p](#)
- int1_p_t [AIRINV::InventoryParserHelper::family_code_p](#)

25.126 InventoryParserHelper.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/Logger.hpp>
00008 #include <stdair/stdair_exceptions.hpp>
00009 // Airinv
00010 #include <airinv/command/InventoryBuilder.hpp>
00011 // #define BOOST_SPIRIT_DEBUG
00012 #include <airinv/command/InventoryParserHelper.hpp>
00013
00014 //
00015 namespace bsc = boost::spirit::classic;
00016
00017 namespace AIRINV {
00018     namespace InventoryParserHelper {
00019
00020
```

```

00021 // //////////////////////////////////////
00022 // Semantic actions
00023 // //////////////////////////////////////
00024
00025 ParserSemanticAction::
00026 ParserSemanticAction (FlightDateStruct& ioFlightDate)
00027 : _flightDate (ioFlightDate) {
00028 }
00029
00030 // //////////////////////////////////////
00031 storeSnapshotDate::
00032 storeSnapshotDate (FlightDateStruct& ioFlightDate)
00033 : ParserSemanticAction (ioFlightDate) {
00034 }
00035
00036 // //////////////////////////////////////
00037 void storeSnapshotDate::operator() (iterator_t iStr,
00038                                     iterator_t iStrEnd) const {
00039     _flightDate._flightDate = _flightDate.getDate();
00040 }
00041
00042 // //////////////////////////////////////
00043 storeAirlineCode::
00044 storeAirlineCode (FlightDateStruct& ioFlightDate)
00045 : ParserSemanticAction (ioFlightDate) {
00046 }
00047
00048 // //////////////////////////////////////
00049 void storeAirlineCode::operator() (iterator_t iStr,
00050                                     iterator_t iStrEnd) const {
00051     const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00052     _flightDate._airlineCode = lAirlineCode;
00053
00054     // As that's the beginning of a new flight, all the list must be reset
00055     // 1. Leg branch of the tree
00056     _flightDate._legList.clear();
00057     _flightDate._itLeg._cabinList.clear();
00058     _flightDate._itLegCabin._bucketList.clear();
00059     _flightDate._itBucket._yieldRangeUpperValue = 0.0;
00060
00061     // 2. Segment branch of the tree
00062     _flightDate._segmentList.clear();
00063     _flightDate._itSegment._cabinList.clear();
00064     _flightDate._itSegmentCabin._itFareFamily._classList.clear();
00065     _flightDate._itSegmentCabin._fareFamilies.clear();
00066     _flightDate._itBookingClass._classCode = "";
00067 }
00068
00069 // //////////////////////////////////////
00070 storeFlightNumber::storeFlightNumber (FlightDateStruct& ioFlightDate)
00071 : ParserSemanticAction (ioFlightDate) {
00072 }
00073
00074 // //////////////////////////////////////
00075 void storeFlightNumber::operator() (unsigned int iNumber) const {
00076     _flightDate._flightNumber = iNumber;
00077 }
00078
00079 // //////////////////////////////////////
00080 storeFlightDate::storeFlightDate (FlightDateStruct& ioFlightDate)
00081 : ParserSemanticAction (ioFlightDate) {
00082 }
00083
00084 // //////////////////////////////////////
00085 void storeFlightDate::operator() (iterator_t iStr,
00086                                     iterator_t iStrEnd) const {
00087     _flightDate._flightDate = _flightDate.getDate();
00088 }
00089
00090 // //////////////////////////////////////
00091 storeFlightTypeCode::storeFlightTypeCode (FlightDateStruct& ioFlightDate)
00092 : ParserSemanticAction (ioFlightDate) {
00093 }
00094
00095 // //////////////////////////////////////
00096 void storeFlightTypeCode::operator() (iterator_t iStr,
00097                                     iterator_t iStrEnd) const {
00098     const std::string lFlightTypeCodeStr (iStr, iStrEnd);
00099     const FlightTypeCode lFlightTypeCode (lFlightTypeCodeStr);
00100     _flightDate._flightTypeCode = lFlightTypeCode.getCode();
00101     //STDAIR_LOG_DEBUG ("FlightType code: " << lFlightTypeCode);
00102 }
00103
00104 // //////////////////////////////////////
00105 storeFlightVisibilityCode::
00106 storeFlightVisibilityCode (FlightDateStruct& ioFlightDate)
00107 : ParserSemanticAction (ioFlightDate) {

```

```

00108     }
00109
00110     // //////////////////////////////////////
00111     void storeFlightVisibilityCode::operator() (iterator_t iStr,
00112                                                iterator_t iStrEnd) const {
00113         const std::string lFlightVisibilityCodeStr (iStr, iStrEnd);
00114         const FlightVisibilityCode lFlightVisibilityCode(lFlightVisibilityCodeStr
);
00115         _flightDate._flightVisibilityCode = lFlightVisibilityCode.getCode();
00116         //STDAIR_LOG_DEBUG ("FlightVisibility code: " << lFlightVisibilityCode);
00117     }
00118
00119     // //////////////////////////////////////
00120     storeLegBoardingPoint::
00121     storeLegBoardingPoint (FlightDateStruct& ioFlightDate)
00122         : ParserSemanticAction (ioFlightDate) {
00123     }
00124
00125     // //////////////////////////////////////
00126     void storeLegBoardingPoint::operator() (iterator_t iStr,
00127                                             iterator_t iStrEnd) const {
00128         stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00129
00130         // //////////////////////////////////
00131         // If this is not the first leg-date of the flight-date,
00132         // the already parsed leg-date must be added to the flight-date.
00133         if (_flightDate._itLeg._cabinList.empty() == false) {
00134             _flightDate._itLegCabin._bucketList.push_back (_flightDate._itBucket);
00135             _flightDate._itLeg._cabinList.push_back (_flightDate._itLegCabin);
00136             _flightDate._legList.push_back (_flightDate._itLeg);
00137         }
00138
00139         // As that's the beginning of a new leg-date,
00140         // (re-)initialise the leg-cabin branch of the tree
00141         _flightDate._itLeg._cabinList.clear();
00142         _flightDate._itLegCabin._cabinCode = "";
00143         _flightDate._itLegCabin._bucketList.clear();
00144         _flightDate._itBucket._yieldRangeUpperValue = 0.0;
00145
00146         // //////////////////////////////////
00147         // Set the (new) boarding point
00148         _flightDate._itLeg._boardingPoint = lBoardingPoint;
00149
00150         // Add the airport code if it is not already stored in the airport lists
00151         _flightDate.addAirport (lBoardingPoint);
00152     }
00153
00154     // //////////////////////////////////////
00155     storeLegOffPoint::
00156     storeLegOffPoint (FlightDateStruct& ioFlightDate)
00157         : ParserSemanticAction (ioFlightDate) {
00158     }
00159
00160     // //////////////////////////////////////
00161     void storeLegOffPoint::operator() (iterator_t iStr,
00162                                       iterator_t iStrEnd) const {
00163         stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00164         _flightDate._itLeg._offPoint = lOffPoint;
00165
00166         // Add the airport code if it is not already stored in the airport lists
00167         _flightDate.addAirport (lOffPoint);
00168     }
00169
00170     // //////////////////////////////////////
00171     storeBoardingDate::storeBoardingDate (FlightDateStruct& ioFlightDate)
00172         : ParserSemanticAction (ioFlightDate) {
00173     }
00174
00175     // //////////////////////////////////////
00176     void storeBoardingDate::operator() (iterator_t iStr,
00177                                       iterator_t iStrEnd) const {
00178         _flightDate._itLeg._boardingDate = _flightDate.getDate();
00179     }
00180
00181     // //////////////////////////////////////
00182     storeBoardingTime::storeBoardingTime (FlightDateStruct& ioFlightDate)
00183         : ParserSemanticAction (ioFlightDate) {
00184     }
00185
00186     // //////////////////////////////////////
00187     void storeBoardingTime::operator() (iterator_t iStr,
00188                                       iterator_t iStrEnd) const {
00189         _flightDate._itLeg._boardingTime = _flightDate.getTime();
00190
00191         // Reset the number of seconds
00192         _flightDate._itSeconds = 0;
00193

```

```

00194
00195     // Reset the date off-set
00196     _flightDate._dateOffset = 0;
00197 }
00198
00199 // //////////////////////////////////////
00200 storeOffDate::storeOffDate (FlightDateStruct& ioFlightDate)
00201 : ParserSemanticAction (ioFlightDate) {
00202 }
00203
00204 // //////////////////////////////////////
00205 void storeOffDate::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00206     _flightDate._itLeg._offDate = _flightDate.getDate();
00207 }
00208
00209 // //////////////////////////////////////
00210 storeOffTime::storeOffTime (FlightDateStruct& ioFlightDate)
00211 : ParserSemanticAction (ioFlightDate) {
00212 }
00213
00214 // //////////////////////////////////////
00215 void storeOffTime::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00216     _flightDate._itLeg._offTime = _flightDate.getTime();
00217 }
00218
00219 // Reset the number of seconds
00220 _flightDate._itSeconds = 0;
00221 }
00222
00223 // //////////////////////////////////////
00224 storeLegCabinCode::storeLegCabinCode (FlightDateStruct& ioFlightDate)
00225 : ParserSemanticAction (ioFlightDate) {
00226 }
00227
00228 // //////////////////////////////////////
00229 void storeLegCabinCode::operator() (char iChar) const {
00230
00231     // //////////////////////////////////
00232     // If this is not the first leg-cabin of the leg-date,
00233     // the already parsed leg-cabin must be added to the leg-date.
00234     if (_flightDate._itLegCabin._cabinCode != "") {
00235         if (_flightDate._itLegCabin._bucketList.empty() == false) {
00236             _flightDate._itLegCabin._bucketList.push_back (_flightDate._itBucket);
00237         }
00238         _flightDate._itLeg._cabinList.push_back (_flightDate._itLegCabin);
00239     }
00240
00241     // (Re-)initialise the leg-cabin branch of the tree
00242     _flightDate._itLegCabin._bucketList.clear();
00243     _flightDate._itBucket._yieldRangeUpperValue = 0.0;
00244
00245     // //////////////////////////////////
00246     _flightDate._itLegCabin._cabinCode = iChar;
00247     //std::cout << "Cabin code: " << iChar << std::endl;
00248 }
00249
00250 // //////////////////////////////////////
00251 storeSaleableCapacity::
00252 storeSaleableCapacity (FlightDateStruct& ioFlightDate)
00253 : ParserSemanticAction (ioFlightDate) {
00254 }
00255
00256 // //////////////////////////////////////
00257 void storeSaleableCapacity::operator() (double iReal) const {
00258     _flightDate._itLegCabin._saleableCapacity = iReal;
00259     //std::cout << "Saleable capacity: " << iReal << std::endl;
00260 }
00261
00262 // //////////////////////////////////////
00263 storeAU::storeAU (FlightDateStruct& ioFlightDate)
00264 : ParserSemanticAction (ioFlightDate) {
00265 }
00266
00267 // //////////////////////////////////////
00268 void storeAU::operator() (double iReal) const {
00269     _flightDate._itLegCabin._au = iReal;
00270     //std::cout << "AU: " << iReal << std::endl;
00271 }
00272
00273 // //////////////////////////////////////
00274 storeUPR::storeUPR (FlightDateStruct& ioFlightDate)
00275 : ParserSemanticAction (ioFlightDate) {
00276 }
00277
00278 // //////////////////////////////////////
00279 void storeUPR::operator() (double iReal) const {

```

```

00280     _flightDate._itLegCabin._upr = iReal;
00281     //std::cout << "UPR: " << iReal << std::endl;
00282 }
00283
00284 // //////////////////////////////////////
00285 storeBookingCounter::storeBookingCounter (FlightDateStruct& ioFlightDate)
00286 : ParserSemanticAction (ioFlightDate) {
00287 }
00288
00289 // //////////////////////////////////////
00290 void storeBookingCounter::operator() (double iReal) const {
00291     _flightDate._itLegCabin._nbOfBookings = iReal;
00292     //std::cout << "Nb of bookings: " << iReal << std::endl;
00293 }
00294
00295 // //////////////////////////////////////
00296 storeNAV::storeNAV (FlightDateStruct& ioFlightDate)
00297 : ParserSemanticAction (ioFlightDate) {
00298 }
00299
00300 // //////////////////////////////////////
00301 void storeNAV::operator() (double iReal) const {
00302     _flightDate._itLegCabin._nav = iReal;
00303     //std::cout << "NAV: " << iReal << std::endl;
00304 }
00305
00306 // //////////////////////////////////////
00307 storeGAV::storeGAV (FlightDateStruct& ioFlightDate)
00308 : ParserSemanticAction (ioFlightDate) {
00309 }
00310
00311 // //////////////////////////////////////
00312 void storeGAV::operator() (double iReal) const {
00313     _flightDate._itLegCabin._gav = iReal;
00314     //std::cout << "GAV: " << iReal << std::endl;
00315 }
00316
00317 // //////////////////////////////////////
00318 storeACP::storeACP (FlightDateStruct& ioFlightDate)
00319 : ParserSemanticAction (ioFlightDate) {
00320 }
00321
00322 // //////////////////////////////////////
00323 void storeACP::operator() (double iReal) const {
00324     _flightDate._itLegCabin._acp = iReal;
00325     //std::cout << "ACP: " << iReal << std::endl;
00326 }
00327
00328 // //////////////////////////////////////
00329 storeETB::storeETB (FlightDateStruct& ioFlightDate)
00330 : ParserSemanticAction (ioFlightDate) {
00331 }
00332
00333 // //////////////////////////////////////
00334 void storeETB::operator() (double iReal) const {
00335     _flightDate._itLegCabin._etb = iReal;
00336     //std::cout << "ETB: " << iReal << std::endl;
00337 }
00338
00339 // //////////////////////////////////////
00340 storeYieldUpperRange::storeYieldUpperRange(FlightDateStruct& ioFlightDate)
00341 : ParserSemanticAction (ioFlightDate) {
00342 }
00343
00344 // //////////////////////////////////////
00345 void storeYieldUpperRange::operator() (double iReal) const {
00346     // If this is not the first bucket of the leg-cabin,
00347     // the already parsed bucket must be added to the leg-cabin.
00348     if (_flightDate._itBucket._yieldRangeUpperValue != 0.0) {
00349         _flightDate._itLegCabin._bucketList.push_back (_flightDate._itBucket);
00350     }
00351
00352
00353     // //////////////////////////////////////
00354     _flightDate._itBucket._yieldRangeUpperValue = iReal;
00355     //std::cout << "Yield Upper Range Value: " << iReal << std::endl;
00356 }
00357
00358 // //////////////////////////////////////
00359 storeBucketAvaibility::
00360 storeBucketAvaibility (FlightDateStruct& ioFlightDate)
00361 : ParserSemanticAction (ioFlightDate) {
00362 }
00363
00364 // //////////////////////////////////////
00365 void storeBucketAvaibility::operator() (double iReal) const {
00366     _flightDate._itBucket._availability = iReal;

```

```

00367         //std::cout << "Availability: " << iReal << std::endl;
00368     }
00369
00370     // //////////////////////////////////////
00371     storeSeatIndex::storeSeatIndex (FlightDateStruct& ioFlightDate)
00372     : ParserSemanticAction (ioFlightDate) {
00373     }
00374
00375     // //////////////////////////////////////
00376     void storeSeatIndex::operator() (double iReal) const {
00377         _flightDate._itBucket._seatIndex = iReal;
00378         //std::cout << "Seat Index: " << iReal << std::endl;
00379     }
00380
00381     // //////////////////////////////////////
00382     storeSegmentBoardingPoint::
00383     storeSegmentBoardingPoint (FlightDateStruct& ioFlightDate)
00384     : ParserSemanticAction (ioFlightDate) {
00385     }
00386
00387     // //////////////////////////////////////
00388     void storeSegmentBoardingPoint::operator() (iterator_t iStr,
00389                                                 iterator_t iStrEnd) const {
00390         stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00391
00392         // //////////////////////////////////////
00393         // When the first segment-date is read, it means that the leg section
00394         // is over. The parsed leg can therefore be added to the list.
00395         if (_flightDate._itLeg._cabinList.empty() == false) {
00396             _flightDate._itLegCabin._bucketList.push_back (_flightDate._itBucket);
00397             _flightDate._itLeg._cabinList.push_back (_flightDate._itLegCabin);
00398             _flightDate._legList.push_back (_flightDate._itLeg);
00399
00400             // (Re-)initialise the leg-date branch of the tree
00401             _flightDate._itLeg._cabinList.clear();
00402             _flightDate._itLegCabin._cabinCode = "";
00403             _flightDate._itLeg._cabinList.clear();
00404             _flightDate._itLegCabin._bucketList.clear();
00405         }
00406
00407         // //////////////////////////////////////
00408         // If this is not the first segment-date of the flight-date,
00409         // the already parsed segment-date must be added to the flight-date.
00410         if (_flightDate._itSegment._cabinList.empty() == false) {
00411             _flightDate._itSegmentCabin._itFareFamily._classList.push_back (
00412                 _flightDate._itBookingClass);
00413             _flightDate._itSegmentCabin._fareFamilies.push_back (_flightDate.
00414                 _itSegmentCabin._itFareFamily);
00415             _flightDate._itSegment._cabinList.push_back (_flightDate._itSegmentCabin
00416 );
00417             _flightDate._segmentList.push_back (_flightDate._itSegment);
00418         }
00419
00420         // As that's the beginning of a new segment-date,
00421         // (re-)initialise the segment-cabin branch of the tree
00422         _flightDate._itSegment._cabinList.clear();
00423         _flightDate._itSegmentCabin._itFareFamily._classList.clear();
00424         _flightDate._itSegmentCabin._fareFamilies.clear();
00425         _flightDate._itBookingClass._classCode = "";
00426
00427         // //////////////////////////////////////
00428         _flightDate._itSegment._boardingPoint = lBoardingPoint;
00429         //std::cout << "Board point: " << lBoardingPoint << std::endl;
00430     }
00431
00432     // //////////////////////////////////////
00433     storeSegmentOffPoint::storeSegmentOffPoint (FlightDateStruct& ioFlightDate)
00434     : ParserSemanticAction (ioFlightDate) {
00435     }
00436
00437     // //////////////////////////////////////
00438     void storeSegmentOffPoint::operator() (iterator_t iStr,
00439                                           iterator_t iStrEnd) const {
00440         stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00441         _flightDate._itSegment._offPoint = lOffPoint;
00442         //std::cout << "Off point: " << lOffPoint << std::endl;
00443     }
00444
00445     // //////////////////////////////////////
00446     storeSegmentCabinCode::
00447     storeSegmentCabinCode (FlightDateStruct& ioFlightDate)
00448     : ParserSemanticAction (ioFlightDate) {
00449     }
00450     // //////////////////////////////////////

```

```

00451 void storeSegmentCabinCode::operator() (char iChar) const {
00452
00453     // Reset the list of fare families, as it is a new segment-cabin
00454     _flightDate._itSegmentCabin._fareFamilies.clear();
00455
00456     // ///////////////////////////////////
00457     // If this is not the first segment-cabin of the segment-date,
00458     // the already parsed segment-cabin must be added to the segment-date.
00459     if (_flightDate._itSegmentCabin._itFareFamily._classList.empty() == false
) {
00460         _flightDate._itSegmentCabin._itFareFamily._classList.
00461             push_back (_flightDate._itBookingClass);
00462         _flightDate._itSegmentCabin._fareFamilies.
00463             push_back (_flightDate._itSegmentCabin._itFareFamily);
00464         _flightDate._itSegment._cabinList.
00465             push_back (_flightDate._itSegmentCabin);
00466     }
00467
00468     // (Re-)initialise the booking-class branch of the tree
00469     _flightDate._itSegmentCabin._fareFamilies.clear();
00470     _flightDate._itSegmentCabin._itFareFamily._classList.clear();
00471     _flightDate._itBookingClass._classCode = "";
00472
00473
00474     // ///////////////////////////////////
00475     _flightDate._itSegmentCabin._cabinCode = iChar;
00476     //std::cout << "Segment-cabin code: " << iChar << std::endl;
00477 }
00478
00479 // ///////////////////////////////////
00480 storeSegmentCabinBookingCounter::
00481 storeSegmentCabinBookingCounter (FlightDateStruct& ioFlightDate)
00482 : ParserSemanticAction (ioFlightDate) {
00483 }
00484
00485 // ///////////////////////////////////
00486 void storeSegmentCabinBookingCounter::operator() (double iReal) const {
00487     _flightDate._itSegmentCabin._nbOfBookings = iReal;
00488     //std::cout << "Nb of bookings: " << iReal << std::endl;
00489 }
00490
00491 // ///////////////////////////////////
00492 storeClassCode::storeClassCode (FlightDateStruct& ioFlightDate)
00493 : ParserSemanticAction (ioFlightDate) {
00494 }
00495
00496 // ///////////////////////////////////
00497 void storeClassCode::operator() (char iChar) const {
00498     // If this is not the first booking-class of the segment-cabin,
00499     // the already parsed booking-class must be added to the segment-cabin.
00500     if (_flightDate._itBookingClass._classCode != "") {
00501         _flightDate._itSegmentCabin._itFareFamily._classList.
00502             push_back (_flightDate._itBookingClass);
00503     }
00504
00505     // ///////////////////////////////////
00506     _flightDate._itBookingClass._classCode = iChar;
00507     //std::cout << "Booking class code: " << iChar << std::endl;
00508 }
00509
00510 // ///////////////////////////////////
00511 storeSubclassCode::storeSubclassCode (FlightDateStruct& ioFlightDate)
00512 : ParserSemanticAction (ioFlightDate) {
00513 }
00514
00515 // ///////////////////////////////////
00516 void storeSubclassCode::operator() (unsigned int iNumber) const {
00517     _flightDate._itBookingClass._subclassCode = iNumber;
00518     //std::cout << "Sub-class code: " << iNumber << std::endl;
00519 }
00520
00521 // ///////////////////////////////////
00522 storeParentClassCode::
00523 storeParentClassCode (FlightDateStruct& ioFlightDate)
00524 : ParserSemanticAction (ioFlightDate) {
00525 }
00526
00527 // ///////////////////////////////////
00528 void storeParentClassCode::operator() (char iChar) const {
00529     _flightDate._itBookingClass._parentClassCode = iChar;
00530     //std::cout << "Parent booking class code: " << iChar << std::endl;
00531 }
00532
00533 // ///////////////////////////////////
00534 storeParentSubclassCode::
00535 storeParentSubclassCode (FlightDateStruct& ioFlightDate)
00536 : ParserSemanticAction (ioFlightDate) {

```



```

00537     }
00538
00539     // //////////////////////////////////////
00540 void storeParentSubclassCode::operator() (unsigned int iNumber) const {
00541     _flightDate._itBookingClass._parentSubclassCode = iNumber;
00542     //std::cout << "Parent sub-class code: " << iNumber << std::endl;
00543 }
00544
00545     // //////////////////////////////////////
00546 storeCumulatedProtection::
00547 storeCumulatedProtection (FlightDateStruct& ioFlightDate)
00548     : ParserSemanticAction (ioFlightDate) {
00549 }
00550
00551     // //////////////////////////////////////
00552 void storeCumulatedProtection::operator() (double iReal) const {
00553     _flightDate._itBookingClass._cumulatedProtection = iReal;
00554     //std::cout << "Cumulated protection: " << iReal << std::endl;
00555 }
00556
00557     // //////////////////////////////////////
00558 storeProtection::storeProtection (FlightDateStruct& ioFlightDate)
00559     : ParserSemanticAction (ioFlightDate) {
00560 }
00561
00562     // //////////////////////////////////////
00563 void storeProtection::operator() (double iReal) const {
00564     _flightDate._itBookingClass._protection = iReal;
00565     //std::cout << "Protection: " << iReal << std::endl;
00566 }
00567
00568     // //////////////////////////////////////
00569 storeNego::storeNego (FlightDateStruct& ioFlightDate)
00570     : ParserSemanticAction (ioFlightDate) {
00571 }
00572
00573     // //////////////////////////////////////
00574 void storeNego::operator() (double iReal) const {
00575     _flightDate._itBookingClass._nego = iReal;
00576     //std::cout << "Negotiated allotment: " << iReal << std::endl;
00577 }
00578
00579     // //////////////////////////////////////
00580 storeNoShow::storeNoShow (FlightDateStruct& ioFlightDate)
00581     : ParserSemanticAction (ioFlightDate) {
00582 }
00583
00584     // //////////////////////////////////////
00585 void storeNoShow::operator() (double iReal) const {
00586     _flightDate._itBookingClass._noShowPercentage = iReal;
00587     //std::cout << "No-Show percentage: " << iReal << std::endl;
00588 }
00589
00590     // //////////////////////////////////////
00591 storeOverbooking::storeOverbooking (FlightDateStruct& ioFlightDate)
00592     : ParserSemanticAction (ioFlightDate) {
00593 }
00594
00595     // //////////////////////////////////////
00596 void storeOverbooking::operator() (double iReal) const {
00597     _flightDate._itBookingClass._overbookingPercentage = iReal;
00598     //std::cout << "Overbooking percentage: " << iReal << std::endl;
00599 }
00600
00601     // //////////////////////////////////////
00602 storeNbOfBkgs::storeNbOfBkgs (FlightDateStruct& ioFlightDate)
00603     : ParserSemanticAction (ioFlightDate) {
00604 }
00605
00606     // //////////////////////////////////////
00607 void storeNbOfBkgs::operator() (double iReal) const {
00608     _flightDate._itBookingClass._nbOfBookings = iReal;
00609     //std::cout << "Nb of bookings: " << iReal << std::endl;
00610 }
00611
00612     // //////////////////////////////////////
00613 storeNbOfGroupBkgs::storeNbOfGroupBkgs (FlightDateStruct& ioFlightDate)
00614     : ParserSemanticAction (ioFlightDate) {
00615 }
00616
00617     // //////////////////////////////////////
00618 void storeNbOfGroupBkgs::operator() (double iReal) const {
00619     _flightDate._itBookingClass._nbOfGroupBookings = iReal;
00620     //std::cout << "Nb of group bookings: " << iReal << std::endl;
00621 }
00622
00623     // //////////////////////////////////////

```

```

00624     storeNbOfPendingGroupBkgs::
00625     storeNbOfPendingGroupBkgs (FlightDateStruct& ioFlightDate)
00626     : ParserSemanticAction (ioFlightDate) {
00627     }
00628
00629     // //////////////////////////////////////
00630     void storeNbOfPendingGroupBkgs::operator() (double iReal) const {
00631         _flightDate._itBookingClass._nbOfPendingGroupBookings = iReal;
00632         //std::cout << "Nb of pending group bookings: " << iReal << std::endl;
00633     }
00634
00635     // //////////////////////////////////////
00636     storeNbOfStaffBkgs::storeNbOfStaffBkgs (FlightDateStruct& ioFlightDate)
00637     : ParserSemanticAction (ioFlightDate) {
00638     }
00639
00640     // //////////////////////////////////////
00641     void storeNbOfStaffBkgs::operator() (double iReal) const {
00642         _flightDate._itBookingClass._nbOfStaffBookings = iReal;
00643         //std::cout << "Nb of staff bookings: " << iReal << std::endl;
00644     }
00645
00646     // //////////////////////////////////////
00647     storeNbOfWLBkgs::storeNbOfWLBkgs (FlightDateStruct& ioFlightDate)
00648     : ParserSemanticAction (ioFlightDate) {
00649     }
00650
00651     // //////////////////////////////////////
00652     void storeNbOfWLBkgs::operator() (double iReal) const {
00653         _flightDate._itBookingClass._nbOfWLBookings = iReal;
00654         //std::cout << "Nb of wait-list bookings: " << iReal << std::endl;
00655     }
00656
00657     // //////////////////////////////////////
00658     storeClassETB::storeClassETB (FlightDateStruct& ioFlightDate)
00659     : ParserSemanticAction (ioFlightDate) {
00660     }
00661
00662     // //////////////////////////////////////
00663     void storeClassETB::operator() (double iReal) const {
00664         _flightDate._itBookingClass._etb = iReal;
00665         //std::cout << "Class-level ETB: " << iReal << std::endl;
00666     }
00667
00668     // //////////////////////////////////////
00669     storeClassAvailability::
00670     storeClassAvailability (FlightDateStruct& ioFlightDate)
00671     : ParserSemanticAction (ioFlightDate) {
00672     }
00673
00674     // //////////////////////////////////////
00675     void storeClassAvailability::operator() (double iReal) const {
00676         _flightDate._itBookingClass._netClassAvailability = iReal;
00677         //std::cout << "Net class availability: " << iReal << std::endl;
00678     }
00679
00680     // //////////////////////////////////////
00681     storeSegmentAvailability::
00682     storeSegmentAvailability (FlightDateStruct& ioFlightDate)
00683     : ParserSemanticAction (ioFlightDate) {
00684     }
00685
00686     // //////////////////////////////////////
00687     void storeSegmentAvailability::operator() (double iReal) const {
00688         _flightDate._itBookingClass._segmentAvailability = iReal;
00689         //std::cout << "Segment availability: " << iReal << std::endl;
00690     }
00691
00692     // //////////////////////////////////////
00693     storeRevenueAvailability::
00694     storeRevenueAvailability (FlightDateStruct& ioFlightDate)
00695     : ParserSemanticAction (ioFlightDate) {
00696     }
00697
00698     // //////////////////////////////////////
00699     void storeRevenueAvailability::operator() (double iReal) const {
00700         _flightDate._itBookingClass._netRevenueAvailability = iReal;
00701         //std::cout << "Net revenue availability: " << iReal << std::endl;
00702     }
00703
00704     // //////////////////////////////////////
00705     storeFamilyCode::storeFamilyCode (FlightDateStruct& ioFlightDate)
00706     : ParserSemanticAction (ioFlightDate) {
00707     }
00708
00709     // //////////////////////////////////////
00710     void storeFamilyCode::operator() (int iCode) const {

```

```

00711     std::ostringstream ostr;
00712     ostr << iCode;
00713     _flightDate._itSegmentCabin._itFareFamily._familyCode = ostr.str();
00714 }
00715
00716 // //////////////////////////////////////
00717 storeFClasses::storeFClasses (FlightDateStruct& ioFlightDate)
00718 : ParserSemanticAction (ioFlightDate) {
00719 }
00720
00721 // //////////////////////////////////////
00722 void storeFClasses::operator() (iterator_t iStr,
00723                                iterator_t iStrEnd) const {
00724     std::string lClasses (iStr, iStrEnd);
00725     _flightDate._itSegmentCabin._itFareFamily._classes = lClasses;
00726
00727     // The list of classes is the last (according to the arrival order
00728     // within the schedule input file) detail of the segment cabin. Hence,
00729     // when a list of classes is parsed, it means that the full segment
00730     // cabin details have already been parsed as well: the segment cabin
00731     // can thus be added to the segment.
00732     _flightDate._itSegmentCabin._itFareFamily._classList.
00733         push_back (_flightDate._itBookingClass);
00734     _flightDate._itSegmentCabin._fareFamilies.
00735         push_back (_flightDate._itSegmentCabin._itFareFamily);
00736     _flightDate._itSegment._cabinList.push_back (_flightDate._itSegmentCabin)
;
00737
00738     // As that's the beginning of a new segment-cabin,
00739     // (re-)initialise the segment-cabin branch of the tree
00740     _flightDate._itSegmentCabin._itFareFamily._classList.clear();
00741     _flightDate._itSegmentCabin._fareFamilies.clear();
00742     _flightDate._itBookingClass._classCode = "";
00743 }
00744
00745 // //////////////////////////////////////
00746 doEndFlightDate::doEndFlightDate (stdair::BomRoot& ioBomRoot,
00747                                   FlightDateStruct& ioFlightDate,
00748                                   unsigned int& ioNbOfFlights)
00749 : ParserSemanticAction (ioFlightDate), _bomRoot (ioBomRoot),
00750   _nbOfFlights (ioNbOfFlights) {
00751 }
00752
00753 // //////////////////////////////////////
00754 // void doEndFlightDate::operator() (char iChar) const {
00755 void doEndFlightDate::operator() (iterator_t iStr,
00756                                   iterator_t iStrEnd) const {
00757
00758     // //////////////////////////////////
00759     // The segment-date section is now over. It means that the
00760     // already parsed segment-date must be added to the flight-date.
00761     if (_flightDate._itSegment._cabinList.empty() == false) {
00762         _flightDate._segmentList.push_back (_flightDate._itSegment);
00763     }
00764
00765     // As that's the beginning of a new flight-date,
00766     // (re-)initialise the segment-cabin branch of the tree
00767     _flightDate._itSegment._cabinList.clear();
00768
00769
00770     // //////////////////////////////////
00771     //if (_nbOfFlights % 1000 == 0) {
00772     //    DEBUG: Display the result
00773     //    STDAIR_LOG_DEBUG ("FlightDate #" << _nbOfFlights
00774     //                     << ": " << _flightDate.describe());
00775     //}
00776
00777     // Build the FlightDate BOM objects
00778     InventoryBuilder::buildInventory (_bomRoot, _flightDate);
00779
00780     //
00781     ++_nbOfFlights;
00782 }
00783
00784 // //////////////////////////////////////
00785 //
00786 // Utility Parsers
00787 //
00788 // //////////////////////////////////////
00789 int1_p_t int1_p;
00790
00791 uint2_p_t uint2_p;
00792
00793 uint1_2_p_t uint1_2_p;
00794
00795 uint1_3_p_t uint1_3_p;

```

```

00801
00803     uint4_p_t uint4_p;
00804
00806     uint1_4_p_t uint1_4_p;
00807
00809     repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3);
00810
00812     bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u);
00813
00815     bounded2_p_t year_p (uint2_p.derived(), 0u, 99u);
00816
00818     bounded2_p_t month_p (uint2_p.derived(), 1u, 12u);
00819
00821     bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00822
00824     repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7);
00825
00827     repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3);
00828
00830     bounded1_2_p_t hours_p (uint1_2_p.derived(), 0u, 24u);
00831
00833     bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u);
00834
00836     bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u);
00837
00839     chset_t cabin_code_p ("A-Z");
00840
00842     chset_t class_code_p ("A-Z");
00843
00845     chset_t passenger_type_p ("A-Z");
00846
00848     int1_p_t family_code_p;
00849
00851     repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26);
00852
00854     bounded1_3_p_t stay_duration_p (uint1_3_p.derived(), 0u, 999u);
00855
00856
00857     // //////////////////////////////////////
00858     // (Boost Spirit) Grammar Definition
00859     // //////////////////////////////////////
00860
00861     // //////////////////////////////////////
00862     InventoryParser::InventoryParser (stdair::BomRoot& ioBomRoot,
00863                                     FlightDateStruct& ioFlightDate,
00864                                     unsigned int& ioNbOfFlights)
00865         : _bomRoot (ioBomRoot), _flightDate (ioFlightDate),
00866           _nbOfFlights (ioNbOfFlights) {
00867     }
00868
00869     // //////////////////////////////////////
00870     template<typename ScannerT>
00871     InventoryParser::definition<ScannerT>::
00872     definition (InventoryParser const& self) {
00873
00874         flight_date_list = *( not_to_be_parsed | flight_date )
00875         ;
00876
00877         not_to_be_parsed =
00878             bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00879                           | bsc::space_p ]
00880         ;
00881
00882         flight_date = flight_key
00883             >> leg_list
00884             >> segment_list
00885             >> flight_date_end[doEndFlightDate (self._bomRoot, self._flightDate,
00886                                                self._nbOfFlights)]
00887         ;
00888
00889         flight_date_end = bsc::ch_p(';')
00890         ;
00891
00892         flight_key = date[storeSnapshotDate(self._flightDate)]
00893             >> '/' >> airline_code
00894             >> '/' >> flight_number
00895             >> '/' >> date[storeFlightDate(self._flightDate)]
00896             >> '/' >> flight_type_code[storeFlightTypeCode(self._flightDate)]
00897             >> !( '/' >> flight_visibility_code[storeFlightVisibilityCode(self.
00898 _flightDate)])
00899         ;
00900
00901         airline_code =
00902             bsc::lexeme_d[ (airline_code_p) [storeAirlineCode(self._flightDate)] ]
00903         ;
00904
00905         flight_number =

```

```

00905         bsc::lexeme_d[ (flight_number_p) [storeFlightNumber (self._flightDate)]]
00906     ;
00907
00908     date =
00909         bsc::lexeme_d[ (day_p) [bsc::assign_a (self._flightDate._itDay) ]
00910             >> (month_p) [bsc::assign_a (self._flightDate._itMonth) ]
00911             >> (year_p) [bsc::assign_a (self._flightDate._itYear) ] ]
00912     ;
00913
00914     flight_type_code =
00915         ( bsc::chseq_p ("INT") | bsc::chseq_p ("DOM") | bsc::chseq_p ("GRD") )
00916     ;
00917
00918     flight_visibility_code =
00919         ( bsc::chseq_p ("HID") | bsc::chseq_p ("PSD") )
00920     ;
00921
00922     leg_list = +( '/' >> leg )
00923     ;
00924
00925     leg = leg_key >> ';' >> leg_details >> leg_cabin_list
00926     ;
00927
00928     leg_key = (airport_p) [storeLegBoardingPoint (self._flightDate)]
00929     >> ';' >> (airport_p) [storeLegOffPoint (self._flightDate)]
00930     ;
00931
00932     leg_details = date[storeBoardingDate (self._flightDate)]
00933     >> ';' >> time[storeBoardingTime (self._flightDate)]
00934     >> ';' >> date[storeOffDate (self._flightDate)]
00935     >> ';' >> time[storeOffTime (self._flightDate)]
00936     ;
00937
00938     leg_cabin_list = +( ';' >> leg_cabin_details >> !bucket_list )
00939     ;
00940
00941     leg_cabin_details = (cabin_code_p) [storeLegCabinCode (self._flightDate)]
00942     >> ',' >> (bsc::ureal_p) [storeSaleableCapacity (self._flightDate)]
00943     >> ',' >> (bsc::real_p) [storeAU (self._flightDate)]
00944     >> ',' >> (bsc::real_p) [storeUPR (self._flightDate)]
00945     >> ',' >> (bsc::real_p) [storeBookingCounter (self._flightDate)]
00946     >> ',' >> (bsc::real_p) [storeNAV (self._flightDate)]
00947     >> ',' >> (bsc::real_p) [storeGAV (self._flightDate)]
00948     >> ',' >> (bsc::ureal_p) [storeACP (self._flightDate)]
00949     >> ',' >> (bsc::real_p) [storeETB (self._flightDate)]
00950     ;
00951
00952     time =
00953         bsc::lexeme_d[
00954             (hours_p) [bsc::assign_a (self._flightDate._itHours) ]
00955             >> (minutes_p) [bsc::assign_a (self._flightDate._itMinutes) ]
00956             >> ! ( (seconds_p) [bsc::assign_a (self._flightDate._itSeconds) ] )
00957         ]
00958     ;
00959
00960     bucket_list = +( ',' >> bucket_details )
00961     ;
00962
00963     bucket_details =
00964         (bsc::ureal_p) [storeYieldUpperRange (self._flightDate)]
00965     >> ':' >> (bsc::real_p) [storeBucketAvaibility (self._flightDate)]
00966     >> ':' >> (uint1_3_p) [storeSeatIndex (self._flightDate)];
00967
00968     segment_list = +( '/' >> segment )
00969     ;
00970
00971     segment = segment_key >> segment_cabin_list
00972     ;
00973
00974     segment_key = (airport_p) [storeSegmentBoardingPoint (self._flightDate)]
00975     >> ';' >> (airport_p) [storeSegmentOffPoint (self._flightDate)]
00976     ;
00977
00978     segment_cabin_list =
00979         +( ';' >> segment_cabin_key >> ','
00980             >> segment_cabin_details >> class_list >> family_cabin_list )
00981     ;
00982
00983     family_cabin_list =
00984         +( ';' >> family_cabin_details)
00985     ;
00986
00987     segment_cabin_key =
00988         (cabin_code_p) [storeSegmentCabinCode (self._flightDate)]
00989     ;
00990
00991     segment_cabin_details =

```

```

00992         (bsc::ureal_p) [storeSegmentCabinBookingCounter (self._flightDate)]
00993     ;
00994
00995     class_list = +( ' ,' >> class_key >> ' |' >> class_details )
00996     ;
00997
00998     class_key = (class_code_p) [storeClassCode (self._flightDate)]
00999     ;
01000
01001     parent_subclass_code =
01002     (class_code_p) [storeParentClassCode (self._flightDate)]
01003     >> (uint1_2_p) [storeParentSubclassCode (self._flightDate)]
01004     ;
01005
01006     class_protection =
01007     (bsc::ureal_p) [storeProtection (self._flightDate)]
01008     ;
01009
01010     class_nego =
01011     (bsc::ureal_p) [storeNego (self._flightDate)]
01012     ;
01013
01014     class_details = (uint1_2_p) [storeSubclassCode (self._flightDate)]
01015     >> ' : ' >> (bsc::ureal_p) [storeCumulatedProtection (self._flightDate)]
01016     >> ' : ' >> !( parent_subclass_code )
01017     >> ' : ' >> !( class_protection )
01018     >> ' : ' >> (bsc::ureal_p) [storeNoShow (self._flightDate)]
01019     >> ' : ' >> (bsc::ureal_p) [storeOverbooking (self._flightDate)]
01020     >> ' : ' >> (bsc::ureal_p) [storeNbOfBkgs (self._flightDate)]
01021     >> ' : ' >> (bsc::ureal_p) [storeNbOfGroupBkgs (self._flightDate)]
01022     >> ' : ' >> (bsc::ureal_p) [storeNbOfPendingGroupBkgs (self._flightDate)]
01023     >> ' : ' >> (bsc::ureal_p) [storeNbOfStaffBkgs (self._flightDate)]
01024     >> ' : ' >> (bsc::ureal_p) [storeNbOfWLBkgs (self._flightDate)]
01025     >> ' : ' >> (bsc::ureal_p) [storeClassETB (self._flightDate)]
01026     >> ' : ' >> !( class_nego )
01027     >> ' : ' >> (bsc::real_p) [storeClassAvailability (self._flightDate)]
01028     >> ' : ' >> (bsc::real_p) [storeSegmentAvailability (self._flightDate)]
01029     >> ' : ' >> (bsc::real_p) [storeRevenueAvailability (self._flightDate)]
01030     ;
01031
01032     family_cabin_details =
01033     (family_code_p) [storeFamilyCode (self._flightDate)]
01034     >> ' ; '
01035     >> (class_code_list_p) [storeFClasses (self._flightDate)]
01036     ;
01037
01038     // BOOST_SPIRIT_DEBUG_NODE (InventoryParser);
01039     BOOST_SPIRIT_DEBUG_NODE (flight_date_list);
01040     BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
01041     BOOST_SPIRIT_DEBUG_NODE (flight_date);
01042     BOOST_SPIRIT_DEBUG_NODE (flight_date_end);
01043     BOOST_SPIRIT_DEBUG_NODE (flight_key);
01044     BOOST_SPIRIT_DEBUG_NODE (airline_code);
01045     BOOST_SPIRIT_DEBUG_NODE (flight_number);
01046     BOOST_SPIRIT_DEBUG_NODE (flight_type_code);
01047     BOOST_SPIRIT_DEBUG_NODE (flight_visibility_code);
01048     BOOST_SPIRIT_DEBUG_NODE (date);
01049     BOOST_SPIRIT_DEBUG_NODE (leg_list);
01050     BOOST_SPIRIT_DEBUG_NODE (leg);
01051     BOOST_SPIRIT_DEBUG_NODE (leg_key);
01052     BOOST_SPIRIT_DEBUG_NODE (leg_details);
01053     BOOST_SPIRIT_DEBUG_NODE (leg_cabin_list);
01054     BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
01055     BOOST_SPIRIT_DEBUG_NODE (bucket_list);
01056     BOOST_SPIRIT_DEBUG_NODE (bucket_details);
01057     BOOST_SPIRIT_DEBUG_NODE (time);
01058     BOOST_SPIRIT_DEBUG_NODE (segment_list);
01059     BOOST_SPIRIT_DEBUG_NODE (segment);
01060     BOOST_SPIRIT_DEBUG_NODE (segment_key);
01061     BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
01062     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_list);
01063     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_key);
01064     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
01065     BOOST_SPIRIT_DEBUG_NODE (class_list);
01066     BOOST_SPIRIT_DEBUG_NODE (class_key);
01067     BOOST_SPIRIT_DEBUG_NODE (parent_subclass_code);
01068     BOOST_SPIRIT_DEBUG_NODE (class_protection);
01069     BOOST_SPIRIT_DEBUG_NODE (class_nego);
01070     BOOST_SPIRIT_DEBUG_NODE (class_details);
01071     BOOST_SPIRIT_DEBUG_NODE (family_cabin_list);
01072     BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
01073 }
01074
01075 // //////////////////////////////////////
01076 template<typename ScannerT>
01077 bsc::rule<ScannerT> const&
01078 InventoryParser::definition<ScannerT>::start () const {

```

```

01079     return flight_date_list;
01080 }
01081 }
01082
01083
01084 //
01085 // Entry class for the file parser
01086 //
01087 //
01088
01089 // //////////////////////////////////////
01090 InventoryFileParser::
01091 InventoryFileParser (stdair::BomRoot& ioBomRoot, const std::string& iFilename
01092 )
01093 : _filename (iFilename), _bomRoot (ioBomRoot),
01094   _nbOfFlights (0) {
01095     init();
01096 }
01097
01098 // //////////////////////////////////////
01099 void InventoryFileParser::init() {
01100     // Open the file
01101     _startIterator = iterator_t (_filename);
01102
01103     // Check the filename exists and can be open
01104     if (!_startIterator) {
01105         std::ostringstream oMessage;
01106         oMessage << "The file " << _filename << " can not be open.";
01107         STDAIR_LOG_ERROR (oMessage.str());
01108         throw InventoryInputFileNotFoundException (oMessage.str());
01109     }
01110
01111     // Create an EOF iterator
01112     _endIterator = _startIterator.make_end();
01113 }
01114
01115 // //////////////////////////////////////
01116 bool InventoryFileParser::buildInventory () {
01117     bool oResult = false;
01118
01119     STDAIR_LOG_DEBUG ("Parsing inventory input file: " << _filename);
01120
01121     // Initialise the parser (grammar) with the helper/staging structure.
01122     InventoryParserHelper::InventoryParser lInventoryParser (_bomRoot,
01123         _flightDate,
01124         _nbOfFlights);
01125
01126     // Launch the parsing of the file and, thanks to the doEndFlightDate
01127     // call-back structure, the building of the whole Inventory BOM
01128     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
01129     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
01130 ,
01131         lInventoryParser,
01132         bsc::space_p - bsc::eol_p);
01133
01134     // Retrieves whether or not the parsing was successful
01135     oResult = info.hit;
01136
01137     const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
01138     if (oResult == true) {
01139         STDAIR_LOG_DEBUG ("Parsing of inventory input file: " << _filename
01140             << " succeeded: read " << info.length
01141             << " characters. The input file has "
01142             << hasBeenFullyReadStr
01143             << "been fully read. Stop point: " << info.stop);
01144     } else {
01145         STDAIR_LOG_ERROR ("Parsing of inventory input file: " << _filename
01146             << " failed: read " << info.length
01147             << " characters. The input file has "
01148             << hasBeenFullyReadStr
01149             << "been fully read. Stop point: " << info.stop);
01150         throw InventoryFileParsingFailedException("Parsing of inventory input
01151 file"
01152             ": " + _filename + " failed");
01153     }
01154     return oResult;
01155 }
01156
01157 }

```

25.127 airinv/command/InventoryParserHelper.hpp File Reference

```
#include <string>
```

```
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FlightDateStruct.hpp>
```

Classes

- struct [AIRINV::InventoryParserHelper::ParserSemanticAction](#)
- struct [AIRINV::InventoryParserHelper::storeSnapshotDate](#)
- struct [AIRINV::InventoryParserHelper::storeAirlineCode](#)
- struct [AIRINV::InventoryParserHelper::storeFlightNumber](#)
- struct [AIRINV::InventoryParserHelper::storeFlightDate](#)
- struct [AIRINV::InventoryParserHelper::storeFlightTypeCode](#)
- struct [AIRINV::InventoryParserHelper::storeFlightVisibilityCode](#)
- struct [AIRINV::InventoryParserHelper::storeLegBoardingPoint](#)
- struct [AIRINV::InventoryParserHelper::storeLegOffPoint](#)
- struct [AIRINV::InventoryParserHelper::storeBoardingDate](#)
- struct [AIRINV::InventoryParserHelper::storeBoardingTime](#)
- struct [AIRINV::InventoryParserHelper::storeOffDate](#)
- struct [AIRINV::InventoryParserHelper::storeOffTime](#)
- struct [AIRINV::InventoryParserHelper::storeLegCabinCode](#)
- struct [AIRINV::InventoryParserHelper::storeSaleableCapacity](#)
- struct [AIRINV::InventoryParserHelper::storeAU](#)
- struct [AIRINV::InventoryParserHelper::storeUPR](#)
- struct [AIRINV::InventoryParserHelper::storeBookingCounter](#)
- struct [AIRINV::InventoryParserHelper::storeNAV](#)
- struct [AIRINV::InventoryParserHelper::storeGAV](#)
- struct [AIRINV::InventoryParserHelper::storeACP](#)
- struct [AIRINV::InventoryParserHelper::storeETB](#)
- struct [AIRINV::InventoryParserHelper::storeYieldUpperRange](#)
- struct [AIRINV::InventoryParserHelper::storeBucketAvaibility](#)
- struct [AIRINV::InventoryParserHelper::storeSeatIndex](#)
- struct [AIRINV::InventoryParserHelper::storeSegmentBoardingPoint](#)
- struct [AIRINV::InventoryParserHelper::storeSegmentOffPoint](#)
- struct [AIRINV::InventoryParserHelper::storeSegmentCabinCode](#)
- struct [AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter](#)
- struct [AIRINV::InventoryParserHelper::storeClassCode](#)
- struct [AIRINV::InventoryParserHelper::storeSubclassCode](#)
- struct [AIRINV::InventoryParserHelper::storeParentClassCode](#)
- struct [AIRINV::InventoryParserHelper::storeParentSubclassCode](#)
- struct [AIRINV::InventoryParserHelper::storeCumulatedProtection](#)
- struct [AIRINV::InventoryParserHelper::storeProtection](#)
- struct [AIRINV::InventoryParserHelper::storeNego](#)
- struct [AIRINV::InventoryParserHelper::storeNoShow](#)
- struct [AIRINV::InventoryParserHelper::storeOverbooking](#)
- struct [AIRINV::InventoryParserHelper::storeNbOfBkgs](#)
- struct [AIRINV::InventoryParserHelper::storeNbOfGroupBkgs](#)
- struct [AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs](#)
- struct [AIRINV::InventoryParserHelper::storeNbOfStaffBkgs](#)
- struct [AIRINV::InventoryParserHelper::storeNbOfWLBkgs](#)
- struct [AIRINV::InventoryParserHelper::storeClassETB](#)
- struct [AIRINV::InventoryParserHelper::storeClassAvailability](#)
- struct [AIRINV::InventoryParserHelper::storeSegmentAvailability](#)

- struct [AIRINV::InventoryParserHelper::storeRevenueAvailability](#)
- struct [AIRINV::InventoryParserHelper::storeFamilyCode](#)
- struct [AIRINV::InventoryParserHelper::storeFClasses](#)
- struct [AIRINV::InventoryParserHelper::doEndFlightDate](#)
- struct [AIRINV::InventoryParserHelper::InventoryParser](#)
- struct [AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >](#)
- class [AIRINV::InventoryFileParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::InventoryParserHelper](#)

25.128 InventoryParserHelper.hpp

```

00001 #ifndef __AIRINV_CMD_INVENTORYPARSERHELPER_HPP
00002 #define __AIRINV_CMD_INVENTORYPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FlightDateStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023     namespace InventoryParserHelper {
00024
00025         // //////////////////////////////////////
00026         // Semantic actions
00027         // //////////////////////////////////////
00029         struct ParserSemanticAction {
00031             ParserSemanticAction (FlightDateStruct&);
00033             FlightDateStruct& _flightDate;
00034         };
00035
00037         struct storeSnapshotDate : public ParserSemanticAction {
00039             storeSnapshotDate (FlightDateStruct&);
00041             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00042         };
00043
00045         struct storeAirlineCode : public ParserSemanticAction {
00047             storeAirlineCode (FlightDateStruct&);
00049             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00050         };
00051
00053         struct storeFlightNumber : public ParserSemanticAction {
00055             storeFlightNumber (FlightDateStruct&);
00057             void operator() (unsigned int iNumber) const;
00058         };
00059
00061         struct storeFlightDate : public ParserSemanticAction {
00063             storeFlightDate (FlightDateStruct&);
00065             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00066         };
00067
00069         struct storeFlightTypeCode : public ParserSemanticAction {
00071             storeFlightTypeCode (FlightDateStruct&);
00073             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00074         };
00075
00077         struct storeFlightVisibilityCode : public ParserSemanticAction {
00079             storeFlightVisibilityCode (FlightDateStruct&);

```

```
00081     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00082 };
00083
00085 struct storeLegBoardingPoint : public ParserSemanticAction {
00087     storeLegBoardingPoint (FlightDateStruct&);
00089     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00090 };
00091
00093 struct storeLegOffPoint : public ParserSemanticAction {
00095     storeLegOffPoint (FlightDateStruct&);
00097     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00098 };
00099
00101 struct storeBoardingDate : public ParserSemanticAction {
00103     storeBoardingDate (FlightDateStruct&);
00105     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00106 };
00107
00109 struct storeBoardingTime : public ParserSemanticAction {
00111     storeBoardingTime (FlightDateStruct&);
00113     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00114 };
00115
00117 struct storeOffDate : public ParserSemanticAction {
00119     storeOffDate (FlightDateStruct&);
00121     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00122 };
00123
00125 struct storeOffTime : public ParserSemanticAction {
00127     storeOffTime (FlightDateStruct&);
00129     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00130 };
00131
00133 struct storeLegCabinCode : public ParserSemanticAction {
00135     storeLegCabinCode (FlightDateStruct&);
00137     void operator() (char iChar) const;
00138 };
00139
00141 struct storeSaleableCapacity : public ParserSemanticAction {
00143     storeSaleableCapacity (FlightDateStruct&);
00145     void operator() (double iReal) const;
00146 };
00147
00149 struct storeAU : public ParserSemanticAction {
00151     storeAU (FlightDateStruct&);
00153     void operator() (double iReal) const;
00154 };
00155
00157 struct storeUPR : public ParserSemanticAction {
00159     storeUPR (FlightDateStruct&);
00161     void operator() (double iReal) const;
00162 };
00163
00165 struct storeBookingCounter : public ParserSemanticAction {
00167     storeBookingCounter (FlightDateStruct&);
00169     void operator() (double iReal) const;
00170 };
00171
00173 struct storeNAV : public ParserSemanticAction {
00175     storeNAV (FlightDateStruct&);
00177     void operator() (double iReal) const;
00178 };
00179
00181 struct storeGAV : public ParserSemanticAction {
00183     storeGAV (FlightDateStruct&);
00185     void operator() (double iReal) const;
00186 };
00187
00189 struct storeACP : public ParserSemanticAction {
00191     storeACP (FlightDateStruct&);
00193     void operator() (double iReal) const;
00194 };
00195
00197 struct storeETB : public ParserSemanticAction {
00199     storeETB (FlightDateStruct&);
00201     void operator() (double iReal) const;
00202 };
00203
00205 struct storeYieldUpperRange : public ParserSemanticAction {
00207     storeYieldUpperRange (FlightDateStruct&);
00209     void operator() (double iReal) const;
00210 };
00211
00213 struct storeBucketAvaibility : public ParserSemanticAction {
00215     storeBucketAvaibility (FlightDateStruct&);
00217     void operator() (double iReal) const;
00218 };
```

```
00219
00221 struct storeSeatIndex : public ParserSemanticAction {
00223     storeSeatIndex (FlightDateStruct&);
00225     void operator() (double iReal) const;
00226 };
00227
00229 struct storeSegmentBoardingPoint : public ParserSemanticAction {
00231     storeSegmentBoardingPoint (FlightDateStruct&);
00233     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00234 };
00235
00237 struct storeSegmentOffPoint : public ParserSemanticAction {
00239     storeSegmentOffPoint (FlightDateStruct&);
00241     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00242 };
00243
00245 struct storeSegmentCabinCode : public ParserSemanticAction {
00247     storeSegmentCabinCode (FlightDateStruct&);
00249     void operator() (char iChar) const;
00250 };
00251
00253 struct storeSegmentCabinBookingCounter : public ParserSemanticAction {
00255     storeSegmentCabinBookingCounter (FlightDateStruct&);
00257     void operator() (double iReal) const;
00258 };
00259
00261 struct storeClassCode : public ParserSemanticAction {
00263     storeClassCode (FlightDateStruct&);
00265     void operator() (char iChar) const;
00266 };
00267
00269 struct storeSubclassCode : public ParserSemanticAction {
00271     storeSubclassCode (FlightDateStruct&);
00273     void operator() (unsigned int iNumber) const;
00274 };
00275
00277 struct storeParentClassCode : public ParserSemanticAction {
00279     storeParentClassCode (FlightDateStruct&);
00281     void operator() (char iChar) const;
00282 };
00283
00285 struct storeParentSubclassCode : public ParserSemanticAction {
00287     storeParentSubclassCode (FlightDateStruct&);
00289     void operator() (unsigned int iNumber) const;
00290 };
00291
00293 struct storeCumulatedProtection : public ParserSemanticAction {
00295     storeCumulatedProtection (FlightDateStruct&);
00297     void operator() (double iReal) const;
00298 };
00299
00301 struct storeProtection : public ParserSemanticAction {
00303     storeProtection (FlightDateStruct&);
00305     void operator() (double iReal) const;
00306 };
00307
00309 struct storeNego : public ParserSemanticAction {
00311     storeNego (FlightDateStruct&);
00313     void operator() (double iReal) const;
00314 };
00315
00317 struct storeNoShow : public ParserSemanticAction {
00319     storeNoShow (FlightDateStruct&);
00321     void operator() (double iReal) const;
00322 };
00323
00325 struct storeOverbooking : public ParserSemanticAction {
00327     storeOverbooking (FlightDateStruct&);
00329     void operator() (double iReal) const;
00330 };
00331
00333 struct storeNbOfBkgs : public ParserSemanticAction {
00335     storeNbOfBkgs (FlightDateStruct&);
00337     void operator() (double iReal) const;
00338 };
00339
00341 struct storeNbOfGroupBkgs : public ParserSemanticAction {
00343     storeNbOfGroupBkgs (FlightDateStruct&);
00345     void operator() (double iReal) const;
00346 };
00347
00349 struct storeNbOfPendingGroupBkgs : public ParserSemanticAction {
00351     storeNbOfPendingGroupBkgs (FlightDateStruct&);
00353     void operator() (double iReal) const;
00354 };
00355
00357 struct storeNbOfStaffBkgs : public ParserSemanticAction {
```

```

00359     storeNbOfStaffBkgs (FlightDateStruct&);
00361     void operator() (double iReal) const;
00362 };
00363
00366 struct storeNbOfWLBkgs : public ParserSemanticAction {
00368     storeNbOfWLBkgs (FlightDateStruct&);
00370     void operator() (double iReal) const;
00371 };
00372
00374 struct storeClassETB : public ParserSemanticAction {
00376     storeClassETB (FlightDateStruct&);
00378     void operator() (double iReal) const;
00379 };
00380
00383 struct storeClassAvailability : public ParserSemanticAction {
00385     storeClassAvailability (FlightDateStruct&);
00387     void operator() (double iReal) const;
00388 };
00389
00392 struct storeSegmentAvailability : public ParserSemanticAction {
00394     storeSegmentAvailability (FlightDateStruct&);
00396     void operator() (double iReal) const;
00397 };
00398
00401 struct storeRevenueAvailability : public ParserSemanticAction {
00403     storeRevenueAvailability (FlightDateStruct&);
00405     void operator() (double iReal) const;
00406 };
00407
00409 struct storeFamilyCode : public ParserSemanticAction {
00411     storeFamilyCode (FlightDateStruct&);
00413     void operator() (int iCode) const;
00414 };
00415
00417 struct storeFClasses : public ParserSemanticAction {
00419     storeFClasses (FlightDateStruct&);
00421     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00422 };
00423
00425 struct doEndFlightDate : public ParserSemanticAction {
00427     doEndFlightDate (stdair::BomRoot&, FlightDateStruct&,
00428                     unsigned int&);
00430     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00432     stdair::BomRoot& _bomRoot;
00433     unsigned int& _nbOfFlights;
00434 };
00435
00436
00438 //
00439 // (Boost Spirit) Grammar Definition
00440 //
00442
00454 struct InventoryParser :
00455     public boost::spirit::classic::grammar<InventoryParser> {
00456
00457     InventoryParser (stdair::BomRoot&, FlightDateStruct&, unsigned int&);
00458
00459     template <typename ScannerT>
00460     struct definition {
00461         definition (InventoryParser const& self);
00462
00463         // Instantiation of rules
00464         boost::spirit::classic::rule<ScannerT> flight_date_list,
00465             not_to_be_parsed,
00466             flight_date, flight_date_end, flight_key, airline_code, flight_number
00467             ,
00468             flight_type_code, flight_visibility_code,
00469             date, leg_list, leg, leg_key, leg_details,
00470             leg_cabin_list, leg_cabin_details,
00471             bucket_list, bucket_details,
00472             time, segment_list, segment, segment_key, full_segment_cabin_details,
00473             segment_cabin_list, segment_cabin_key, segment_cabin_details,
00474             class_list, class_key, parent_subclass_code,
00475             class_protection, class_nego, class_details,
00476             family_cabin_list, family_cabin_details;
00477
00478         boost::spirit::classic::rule<ScannerT> const& start() const;
00479     };
00480
00481     // Parser Context
00482     stdair::BomRoot& _bomRoot;
00483     FlightDateStruct& _flightDate;
00484     unsigned int& _nbOfFlights;
00485 };
00486
00487 }
00488

```

```

00489
00491 //
00492 // Entry class for the file parser
00493 //
00495
00500 class InventoryFileParser : public stdair::CmdAbstract {
00501 public:
00503     InventoryFileParser (stdair::BomRoot&,
00504                          const stdair::Filename_T& iInventoryInputFilename);
00505
00507     bool buildInventory ();
00508
00509 private:
00511     void init();
00512
00513 private:
00514     // Attributes
00516     stdair::Filename_T _filename;
00517
00519     iterator_t _startIterator;
00520
00522     iterator_t _endIterator;
00523
00525     stdair::BomRoot& _bomRoot;
00526
00528     FlightDateStruct _flightDate;
00529
00531     unsigned int _nbOfFlights;
00532 };
00533
00534 }
00535 #endif // __AIRINV_CMD_INVENTORYPARSERHELPER_HPP

```

25.129 airinv/command/ScheduleParser.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/ScheduleParserHelper.hpp>
#include <airinv/command/ScheduleParser.hpp>
#include <airinv/command/InventoryManager.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.130 ScheduleParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/ScheduleParserHelper.hpp>
00013 #include <airinv/command/ScheduleParser.hpp>
00014 #include <airinv/command/InventoryManager.hpp>
00015
00016 namespace AIRINV {
00017
00018 // //////////////////////////////////////
00019 void ScheduleParser::
00020 generateInventories (const stdair::Filename_T& iScheduleFilename,
00021                    stdair::BomRoot& ioBomRoot) {
00022
00023     // Check that the file path given as input corresponds to an actual file

```

```

00024     bool doesExistAndIsReadable =
00025         stdair::BasFileMgr::doesExistAndIsReadable (iScheduleFilename);
00026     if (doesExistAndIsReadable == false) {
00027         std::ostringstream oMessage;
00028         oMessage << "The schedule input file, '" << iScheduleFilename
00029             << "', can not be retrieved on the file-system";
00030         STDAIR_LOG_ERROR (oMessage.str());
00031         throw ScheduleInputFileNotFoundException (oMessage.str());
00032     }
00033
00034     // Initialise the Flight-Period file parser.
00035     FlightPeriodFileParser lFlightPeriodParser (ioBomRoot, iScheduleFilename);
00036
00037     // Parse the CSV-formatted schedule input file, and generate the
00038     // corresponding Inventories for the airlines.
00039     lFlightPeriodParser.generateInventories ();
00040
00041     // Complete the BomRoot BOM building
00042     // Create the routings for all the inventories.
00043     InventoryManager::createDirectAccesses (ioBomRoot);
00044
00045     // Build the similar flight-date sets and the corresponding guillotine
00046     // blocks.
00047     InventoryManager::buildSimilarSegmentCabinSets (ioBomRoot);
00048
00049     // Bid price vector initialisation
00050     InventoryManager::setDefaultBidPriceVector (ioBomRoot);
00051
00052 }
00053
00054 }

```

25.131 airinv/command/ScheduleParser.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [AIRINV::ScheduleParser](#)
Class wrapping the parser entry point.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.132 ScheduleParser.hpp

```

00001 #ifndef __AIRINV_CMD_SCHEDULEPARSER_HPP
00002 #define __AIRINV_CMD_SCHEDULEPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00012 namespace stdair {
00013     class BomRoot;
00014 }
00015
00016 namespace AIRINV {
00017
00021     class ScheduleParser : public stdair::CmdAbstract {
00022     public:
00031         static void generateInventories (const stdair::Filename_T&
iScheduleFilename,
00032                                     stdair::BomRoot&);

```

```

00033     };
00034 }
00035 #endif // __AIRINV_CMD_SCHEDULEPARSER_HPP

```

25.133 airinv/command/ScheduleParserHelper.cpp File Reference

```

#include <cassert>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/InventoryGenerator.hpp>
#include <airinv/command/ScheduleParserHelper.hpp>

```

Namespaces

- namespace [AIRINV](#)
- namespace [AIRINV::ScheduleParserHelper](#)

Functions

- [repeat_p_t AIRINV::ScheduleParserHelper::airline_code_p](#) (chset_t("0-9A-Z").derived(), 2, 3)
- [bounded1_4_p_t AIRINV::ScheduleParserHelper::flight_number_p](#) (uint1_4_p.derived(), 0u, 9999u)
- [bounded4_p_t AIRINV::ScheduleParserHelper::year_p](#) (uint4_p.derived(), 2000u, 2099u)
- [bounded2_p_t AIRINV::ScheduleParserHelper::month_p](#) (uint2_p.derived(), 1u, 12u)
- [bounded2_p_t AIRINV::ScheduleParserHelper::day_p](#) (uint2_p.derived(), 1u, 31u)
- [repeat_p_t AIRINV::ScheduleParserHelper::dow_p](#) (chset_t("0-1").derived().derived(), 7, 7)
- [repeat_p_t AIRINV::ScheduleParserHelper::airport_p](#) (chset_t("0-9A-Z").derived(), 3, 3)
- [bounded2_p_t AIRINV::ScheduleParserHelper::hours_p](#) (uint2_p.derived(), 0u, 23u)
- [bounded2_p_t AIRINV::ScheduleParserHelper::minutes_p](#) (uint2_p.derived(), 0u, 59u)
- [bounded2_p_t AIRINV::ScheduleParserHelper::seconds_p](#) (uint2_p.derived(), 0u, 59u)
- [chset_t AIRINV::ScheduleParserHelper::cabin_code_p](#) ("A-Z")
- [repeat_p_t AIRINV::ScheduleParserHelper::class_code_list_p](#) (chset_t("A-Z").derived(), 1, 26)

Variables

- [int1_p_t AIRINV::ScheduleParserHelper::int1_p](#)
- [uint2_p_t AIRINV::ScheduleParserHelper::uint2_p](#)
- [uint4_p_t AIRINV::ScheduleParserHelper::uint4_p](#)
- [uint1_4_p_t AIRINV::ScheduleParserHelper::uint1_4_p](#)
- [int1_p_t AIRINV::ScheduleParserHelper::family_code_p](#)

25.134 ScheduleParserHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_exceptions.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AirInv
00011 #include <airinv/command/InventoryGenerator.hpp>
00012 // #define BOOST_SPIRIT_DEBUG
00013 #include <airinv/command/ScheduleParserHelper.hpp>
00014
00015 //

```

```

00016 namespace bsc = boost::spirit::classic;
00017
00018 namespace AIRINV {
00019
00020     namespace ScheduleParserHelper {
00021
00022         // //////////////////////////////////////
00023         // Semantic actions
00024         // //////////////////////////////////////
00025
00026         ParserSemanticAction::
00027         ParserSemanticAction (FlightPeriodStruct& ioFlightPeriod)
00028             : _flightPeriod (ioFlightPeriod) {
00029         }
00030
00031         // //////////////////////////////////////
00032         storeAirlineCode::
00033         storeAirlineCode (FlightPeriodStruct& ioFlightPeriod)
00034             : ParserSemanticAction (ioFlightPeriod) {
00035         }
00036
00037         // //////////////////////////////////////
00038         void storeAirlineCode::operator() (iterator_t iStr,
00039                                           iterator_t iStrEnd) const {
00040             const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00041             _flightPeriod._airlineCode = lAirlineCode;
00042
00043             // As that's the beginning of a new flight, the list of legs
00044             // must be reset
00045             _flightPeriod._legList.clear();
00046         }
00047
00048         // //////////////////////////////////////
00049         storeFlightNumber::
00050         storeFlightNumber (FlightPeriodStruct& ioFlightPeriod)
00051             : ParserSemanticAction (ioFlightPeriod) {
00052         }
00053
00054         // //////////////////////////////////////
00055         void storeFlightNumber::operator() (unsigned int iNumber) const {
00056             _flightPeriod._flightNumber = iNumber;
00057         }
00058
00059         // //////////////////////////////////////
00060         storeDateRangeStart::
00061         storeDateRangeStart (FlightPeriodStruct& ioFlightPeriod)
00062             : ParserSemanticAction (ioFlightPeriod) {
00063         }
00064
00065         // //////////////////////////////////////
00066         void storeDateRangeStart::operator() (iterator_t iStr,
00067                                               iterator_t iStrEnd) const {
00068             _flightPeriod._dateRangeStart = _flightPeriod.getDate();
00069
00070             // Reset the number of seconds
00071             _flightPeriod._itSeconds = 0;
00072         }
00073
00074         // //////////////////////////////////////
00075         storeDateRangeEnd::
00076         storeDateRangeEnd (FlightPeriodStruct& ioFlightPeriod)
00077             : ParserSemanticAction (ioFlightPeriod) {
00078         }
00079
00080         // //////////////////////////////////////
00081         void storeDateRangeEnd::operator() (iterator_t iStr,
00082                                             iterator_t iStrEnd) const {
00083             // As a Boost date period (DatePeriod_T) defines the last day of
00084             // the period to be end-date - one day, we have to add one day to that
00085             // end date before.
00086             const stdair::DateOffset_T oneDay (1);
00087             _flightPeriod._dateRangeEnd = _flightPeriod.getDate() + oneDay;
00088
00089             // Transform the date pair (i.e., the date range) into a date period
00090             _flightPeriod._dateRange =
00091                 stdair::DatePeriod_T (_flightPeriod._dateRangeStart,
00092                                       _flightPeriod._dateRangeEnd);
00093
00094             // Reset the number of seconds
00095             _flightPeriod._itSeconds = 0;
00096         }
00097
00098         // //////////////////////////////////////
00099         storeDow::storeDow (FlightPeriodStruct& ioFlightPeriod)
00100             : ParserSemanticAction (ioFlightPeriod) {
00101         }
00102

```



```

00103 // //////////////////////////////////////
00104 void storeDow::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00105     stdair::DOW_String_T lDow (iStr, iStrEnd);
00106     _flightPeriod._dow = lDow;
00107 }
00108 // //////////////////////////////////////
00109 storeLegBoardingPoint::
00110 storeLegBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00111 : ParserSemanticAction (ioFlightPeriod) {
00112 }
00113 // //////////////////////////////////////
00114 void storeLegBoardingPoint::operator() (iterator_t iStr,
00115     iterator_t iStrEnd) const {
00116     stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00117
00118     // If a leg has already been parsed, add it to the FlightPeriod
00119     if (_flightPeriod._legAlreadyDefined == true) {
00120         _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00121     } else {
00122         _flightPeriod._legAlreadyDefined = true;
00123     }
00124
00125     // Set the (new) boarding point
00126     _flightPeriod._itLeg._boardingPoint = lBoardingPoint;
00127
00128     // As that's the beginning of a new leg, the list of cabins
00129     // must be reset
00130     _flightPeriod._itLeg._cabinList.clear();
00131
00132     // Add the airport code if it is not already stored in the airport lists
00133     _flightPeriod.addAirport (lBoardingPoint);
00134 }
00135 // //////////////////////////////////////
00136 storeLegOffPoint::
00137 storeLegOffPoint (FlightPeriodStruct& ioFlightPeriod)
00138 : ParserSemanticAction (ioFlightPeriod) {
00139 }
00140 // //////////////////////////////////////
00141 void storeLegOffPoint::operator() (iterator_t iStr,
00142     iterator_t iStrEnd) const {
00143     stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00144     _flightPeriod._itLeg._offPoint = lOffPoint;
00145
00146     // Add the airport code if it is not already stored in the airport lists
00147     _flightPeriod.addAirport (lOffPoint);
00148 }
00149 // //////////////////////////////////////
00150 storeBoardingTime::
00151 storeBoardingTime (FlightPeriodStruct& ioFlightPeriod)
00152 : ParserSemanticAction (ioFlightPeriod) {
00153 }
00154 // //////////////////////////////////////
00155 void storeBoardingTime::operator() (iterator_t iStr,
00156     iterator_t iStrEnd) const {
00157     _flightPeriod._itLeg._boardingTime = _flightPeriod.getTime();
00158
00159     // Reset the number of seconds
00160     _flightPeriod._itSeconds = 0;
00161
00162     // Reset the date off-set
00163     _flightPeriod._dateOffset = 0;
00164 }
00165 // //////////////////////////////////////
00166 storeOffTime::
00167 storeOffTime (FlightPeriodStruct& ioFlightPeriod)
00168 : ParserSemanticAction (ioFlightPeriod) {
00169 }
00170 // //////////////////////////////////////
00171 void storeOffTime::operator() (iterator_t iStr,
00172     iterator_t iStrEnd) const {
00173     _flightPeriod._itLeg._offTime = _flightPeriod.getTime();
00174
00175     // Reset the number of seconds
00176     _flightPeriod._itSeconds = 0;
00177
00178     // As the boarding date off set is optional, it can be set only
00179     // afterwards, based on the staging date off-set value
00180     // (_flightPeriod._dateOffset).
00181     const stdair::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00182

```

```

00190     _flightPeriod._itLeg._boardingDateOffset = lDateOffset;
00191 }
00192
00193 // //////////////////////////////////////
00194 storeElapsedTime::
00195 storeElapsedTime (FlightPeriodStruct& ioFlightPeriod)
00196 : ParserSemanticAction (ioFlightPeriod) {
00197 }
00198
00199 // //////////////////////////////////////
00200 void storeElapsedTime::operator() (iterator_t iStr,
00201                                   iterator_t iStrEnd) const {
00202     _flightPeriod._itLeg._elapsed = _flightPeriod.getTime();
00203
00204     // Reset the number of seconds
00205     _flightPeriod._itSeconds = 0;
00206
00207     // As the boarding date off set is optional, it can be set only
00208     // afterwards, based on the staging date off-set value
00209     // (_flightPeriod._dateOffset).
00210     const std::air::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00211     _flightPeriod._itLeg._offDateOffset = lDateOffset;
00212 }
00213
00214 // //////////////////////////////////////
00215 storeLegCabinCode::
00216 storeLegCabinCode (FlightPeriodStruct& ioFlightPeriod)
00217 : ParserSemanticAction (ioFlightPeriod) {
00218 }
00219
00220 // //////////////////////////////////////
00221 void storeLegCabinCode::operator() (char iChar) const {
00222     _flightPeriod._itLegCabin._cabinCode = iChar;
00223     //std::cout << "Cabin code: " << iChar << std::endl;
00224 }
00225
00226 // //////////////////////////////////////
00227 storeCapacity::
00228 storeCapacity (FlightPeriodStruct& ioFlightPeriod)
00229 : ParserSemanticAction (ioFlightPeriod) {
00230 }
00231
00232 // //////////////////////////////////////
00233 void storeCapacity::operator() (double iReal) const {
00234     _flightPeriod._itLegCabin._saleableCapacity = iReal;
00235     //std::cout << "Capacity: " << iReal << std::endl;
00236
00237     // The capacity is the last (according to the arrival order
00238     // within the schedule input file) detail of the leg cabin. Hence,
00239     // when a capacity is parsed, it means that the full cabin
00240     // details have already been parsed as well: the cabin can
00241     // thus be added to the leg.
00242     _flightPeriod._itLeg._cabinList.push_back (_flightPeriod._itLegCabin);
00243 }
00244
00245 // //////////////////////////////////////
00246 storeSegmentSpecificity::
00247 storeSegmentSpecificity (FlightPeriodStruct& ioFlightPeriod)
00248 : ParserSemanticAction (ioFlightPeriod) {
00249 }
00250
00251 // //////////////////////////////////////
00252 void storeSegmentSpecificity::operator() (char iChar) const {
00253     if (iChar == '0') {
00254         _flightPeriod._areSegmentDefinitionsSpecific = false;
00255     } else {
00256         _flightPeriod._areSegmentDefinitionsSpecific = true;
00257     }
00258
00259     // Do a few sanity checks: the two lists should get exactly the same
00260     // content (in terms of airport codes). The only difference is that one
00261     // is a STL set, and the other a STL vector.
00262     assert (_flightPeriod._airportList.size()
00263           == _flightPeriod._airportOrderedList.size());
00264     assert (_flightPeriod._airportList.size() >= 2);
00265
00266     // Since all the legs have now been parsed, we get all the airports
00267     // and the segments may be built.
00268     _flightPeriod.buildSegments();
00269 }
00270
00271 // //////////////////////////////////////
00272 storeSegmentBoardingPoint::
00273 storeSegmentBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00274 : ParserSemanticAction (ioFlightPeriod) {
00275 }
00276

```

```

00277 // //////////////////////////////////////
00278 void storeSegmentBoardingPoint::operator() (iterator_t iStr,
00279                                           iterator_t iStrEnd) const {
00280     stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00281     _flightPeriod._itSegment._boardingPoint = lBoardingPoint;
00282 }
00283
00284 // //////////////////////////////////////
00285 storeSegmentOffPoint::
00286 storeSegmentOffPoint (FlightPeriodStruct& ioFlightPeriod)
00287     : ParserSemanticAction (ioFlightPeriod) {
00288 }
00289
00290 // //////////////////////////////////////
00291 void storeSegmentOffPoint::operator() (iterator_t iStr,
00292                                       iterator_t iStrEnd) const {
00293     stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00294     _flightPeriod._itSegment._offPoint = lOffPoint;
00295 }
00296
00297 // //////////////////////////////////////
00298 storeSegmentCabinCode::
00299 storeSegmentCabinCode (FlightPeriodStruct& ioFlightPeriod)
00300     : ParserSemanticAction (ioFlightPeriod) {
00301 }
00302
00303 // //////////////////////////////////////
00304 void storeSegmentCabinCode::operator() (char iChar) const {
00305     _flightPeriod._itSegmentCabin._cabinCode = iChar;
00306 }
00307
00308 // //////////////////////////////////////
00309 storeClasses::
00310 storeClasses (FlightPeriodStruct& ioFlightPeriod)
00311     : ParserSemanticAction (ioFlightPeriod) {
00312 }
00313
00314 // //////////////////////////////////////
00315 void storeClasses::operator() (iterator_t iStr,
00316                               iterator_t iStrEnd) const {
00317     std::string lClasses (iStr, iStrEnd);
00318     _flightPeriod._itSegmentCabin._itFareFamily._classes = lClasses;
00319
00320     // The list of classes is the last (according to the arrival order
00321     // within the schedule input file) detail of the segment cabin. Hence,
00322     // when a list of classes is parsed, it means that the full segment
00323     // cabin details have already been parsed as well: the segment cabin
00324     // can thus be added to the segment.
00325     if (_flightPeriod._areSegmentDefinitionsSpecific == true) {
00326         _flightPeriod.addSegmentCabin (_flightPeriod._itSegment,
00327                                       _flightPeriod._itSegmentCabin);
00328     } else {
00329         _flightPeriod.addSegmentCabin (_flightPeriod._itSegmentCabin);
00330     }
00331 }
00332
00333 // //////////////////////////////////////
00334 storeFamilyCode::
00335 storeFamilyCode (FlightPeriodStruct& ioFlightPeriod)
00336     : ParserSemanticAction (ioFlightPeriod) {
00337 }
00338
00339 // //////////////////////////////////////
00340 void storeFamilyCode::operator() (int iCode) const {
00341     std::ostringstream ostr;
00342     ostr << iCode;
00343     _flightPeriod._itSegmentCabin._itFareFamily._familyCode = ostr.str();
00344 }
00345
00346 // //////////////////////////////////////
00347 storeFClasses::
00348 storeFClasses (FlightPeriodStruct& ioFlightPeriod)
00349     : ParserSemanticAction (ioFlightPeriod) {
00350 }
00351
00352 // //////////////////////////////////////
00353 void storeFClasses::operator() (iterator_t iStr,
00354                               iterator_t iStrEnd) const {
00355     std::string lClasses (iStr, iStrEnd);
00356     FareFamilyStruct lFareFamily (_flightPeriod._itSegmentCabin._itFareFamily
00357     ._familyCode,
00358                                lClasses);
00359
00360     // The list of classes is the last (according to the arrival order
00361     // within the schedule input file) detail of the segment cabin. Hence,
00362     // when a list of classes is parsed, it means that the full segment
00363     // cabin details have already been parsed as well: the segment cabin

```

```

00363         // can thus be added to the segment.
00364         if (_flightPeriod._areSegmentDefinitionsSpecific == true) {
00365             _flightPeriod.addFareFamily (_flightPeriod._itSegment,
00366                                         _flightPeriod._itSegmentCabin,
00367                                         lFareFamily);
00368         } else {
00369             _flightPeriod.addFareFamily (_flightPeriod._itSegmentCabin,
00370                                         lFareFamily);
00371         }
00372     }
00373
00374     // //////////////////////////////////////
00375     doEndFlight::
00376     doEndFlight (stdair::BomRoot& ioBomRoot,
00377                 FlightPeriodStruct& ioFlightPeriod)
00378         : ParserSemanticAction (ioFlightPeriod),
00379           _bomRoot (ioBomRoot) {
00380     }
00381
00382     // //////////////////////////////////////
00383     // void doEndFlight::operator() (char iChar) const {
00384     void doEndFlight::operator() (iterator_t iStr,
00385                                   iterator_t iStrEnd) const {
00386
00387         assert (_flightPeriod._legAlreadyDefined == true);
00388         _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00389
00390         // The lists of legs and cabins must be reset
00391         _flightPeriod._legAlreadyDefined = false;
00392         _flightPeriod._itLeg._cabinList.clear();
00393
00394         // DEBUG: Display the result
00395         STDAIR_LOG_DEBUG ("FlightPeriod: " << _flightPeriod.describe());
00396
00397         // Create the FlightDate BOM objects, and potentially the intermediary
00398         // objects (e.g., Inventory).
00399         InventoryGenerator::createFlightDate (_bomRoot, _flightPeriod);
00400     }
00401
00402
00403     // //////////////////////////////////////
00404     //
00405     // Utility Parsers
00406     //
00407     // //////////////////////////////////////
00408     int1_p_t int1_p;
00409
00410     uint2_p_t uint2_p;
00411
00412     uint4_p_t uint4_p;
00413
00414     uint1_4_p_t uint1_4_p;
00415
00416     repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3);
00417
00418     bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u);
00419
00420     bounded4_p_t year_p (uint4_p.derived(), 2000u, 2099u);
00421
00422     bounded2_p_t month_p (uint2_p.derived(), 1u, 12u);
00423
00424     bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00425
00426     repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7);
00427
00428     repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3);
00429
00430     bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u);
00431
00432     bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u);
00433
00434     bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u);
00435
00436     chset_t cabin_code_p ("A-Z");
00437
00438     int1_p_t family_code_p;
00439
00440     repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26);
00441
00442     // //////////////////////////////////////
00443     // (Boost Spirit) Grammar Definition
00444     // //////////////////////////////////////
00445     // //////////////////////////////////////
00446     FlightPeriodParser::
00447     FlightPeriodParser (stdair::BomRoot& ioBomRoot,

```

```

00467         FlightPeriodStruct& ioFlightPeriod)
00468     : _bomRoot (ioBomRoot),
00469       _flightPeriod (ioFlightPeriod) {
00470 }
00471
00472 // //////////////////////////////////////
00473 template<typename ScannerT>
00474 FlightPeriodParser::definition<ScannerT>::
00475 definition (FlightPeriodParser const& self) {
00476     flight_period_list = *( not_to_be_parsed | flight_period )
00477     ;
00478
00479     not_to_be_parsed =
00480         bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00481             | bsc::space_p ]
00482         ;
00483
00484     flight_period = flight_key
00485         >> +( ';>' >> leg )
00486         >> ';>' >> segment_section
00487         >> flight_period_end[doEndFlight (self._bomRoot, self._flightPeriod)]
00488         ;
00489
00490     flight_period_end = bsc::ch_p(';>')
00491     ;
00492
00493     flight_key = airline_code
00494         >> ';>' >> flight_number
00495         >> ';>' >> date[storeDateRangeStart (self._flightPeriod)]
00496         >> ';>' >> date[storeDateRangeEnd (self._flightPeriod)]
00497         >> ';>' >> dow[storeDow (self._flightPeriod)]
00498         ;
00499
00500     airline_code =
00501         bsc::lexeme_d[ (airline_code_p) [storeAirlineCode (self._flightPeriod)]]
00502         ;
00503
00504     flight_number =
00505         bsc::lexeme_d[ (flight_number_p) [storeFlightNumber (self._flightPeriod)]]
00506         ;
00507
00508     date =
00509         bsc::lexeme_d[ (year_p) [bsc::assign_a (self._flightPeriod._itYear)]
00510             >> '->' >> (month_p) [bsc::assign_a (self._flightPeriod._itMonth)]
00511             >> '->' >> (day_p) [bsc::assign_a (self._flightPeriod._itDay)]]
00512         ;
00513
00514     dow = bsc::lexeme_d[ dow_p ]
00515     ;
00516
00517     leg = leg_key >> ';>' >> leg_details >> +( ';>' >> leg_cabin_details )
00518     ;
00519
00520     leg_key =
00521         (airport_p) [storeLegBoardingPoint (self._flightPeriod)]
00522         >> ';>'
00523         >> (airport_p) [storeLegOffPoint (self._flightPeriod)]
00524         ;
00525
00526     leg_details =
00527         time[storeBoardingTime (self._flightPeriod)]
00528         >> !(date_offset)
00529         >> ';>'
00530         >> time[storeOffTime (self._flightPeriod)]
00531         >> !(date_offset)
00532         >> ';>'
00533         >> time[storeElapsedTime (self._flightPeriod)]
00534         ;
00535
00536     time =
00537         bsc::lexeme_d[ (hours_p) [bsc::assign_a (self._flightPeriod._itHours)]
00538             >> ':>' >> (minutes_p) [bsc::assign_a (self._flightPeriod._itMinutes)]
00539             >> !(':>' >> (seconds_p) [bsc::assign_a (self._flightPeriod._itSeconds)]
00540         )]
00541     ;
00542
00543     date_offset =
00544         bsc::ch_p(';>')
00545         >> (intl_p) [bsc::assign_a (self._flightPeriod._dateOffset)]
00546         ;
00547
00548     leg_cabin_details =
00549         (cabin_code_p) [storeLegCabinCode (self._flightPeriod)]
00550         >> ';>' >> (bsc::ureal_p) [storeCapacity (self._flightPeriod)]
00551         ;
00552

```

```

00553     segment_key =
00554         (airport_p)[storeSegmentBoardingPoint(self._flightPeriod)]
00555         >> ';'
00556         >> (airport_p)[storeSegmentOffPoint(self._flightPeriod)]
00557     ;
00558
00559     segment_section =
00560         generic_segment | specific_segment_list
00561     ;
00562
00563     generic_segment =
00564         bsc::ch_p('0')[storeSegmentSpecificity(self._flightPeriod)]
00565         >> +(';' >> segment_cabin_details)
00566     ;
00567
00568     specific_segment_list =
00569         bsc::ch_p('1')[storeSegmentSpecificity(self._flightPeriod)]
00570         >> +(';' >> segment_key >> full_segment_cabin_details)
00571     ;
00572
00573     full_segment_cabin_details =
00574         +(';' >> segment_cabin_details)
00575     ;
00576
00577     segment_cabin_details =
00578         (cabin_code_p)[storeSegmentCabinCode(self._flightPeriod)]
00579         >> ';' >> (class_code_list_p)[storeClasses(self._flightPeriod)]
00580         >> +(';' >> family_cabin_details)
00581     ;
00582
00583     family_cabin_details =
00584         (family_code_p)[storeFamilyCode(self._flightPeriod)]
00585         >> ';'
00586         >> (class_code_list_p)[storeFClasses(self._flightPeriod)]
00587     ;
00588
00589     // BOOST_SPIRIT_DEBUG_NODE (FlightPeriodParser);
00590     BOOST_SPIRIT_DEBUG_NODE (flight_period_list);
00591     BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
00592     BOOST_SPIRIT_DEBUG_NODE (flight_period);
00593     BOOST_SPIRIT_DEBUG_NODE (flight_period_end);
00594     BOOST_SPIRIT_DEBUG_NODE (flight_key);
00595     BOOST_SPIRIT_DEBUG_NODE (airline_code);
00596     BOOST_SPIRIT_DEBUG_NODE (flight_number);
00597     BOOST_SPIRIT_DEBUG_NODE (date);
00598     BOOST_SPIRIT_DEBUG_NODE (dow);
00599     BOOST_SPIRIT_DEBUG_NODE (leg);
00600     BOOST_SPIRIT_DEBUG_NODE (leg_key);
00601     BOOST_SPIRIT_DEBUG_NODE (leg_details);
00602     BOOST_SPIRIT_DEBUG_NODE (time);
00603     BOOST_SPIRIT_DEBUG_NODE (date_offset);
00604     BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
00605     BOOST_SPIRIT_DEBUG_NODE (segment_section);
00606     BOOST_SPIRIT_DEBUG_NODE (segment_key);
00607     BOOST_SPIRIT_DEBUG_NODE (generic_segment);
00608     BOOST_SPIRIT_DEBUG_NODE (specific_segment_list);
00609     BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
00610     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
00611     BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
00612 }
00613
00614 // //////////////////////////////////////
00615 template<typename ScannerT>
00616 bsc::rule<ScannerT> const&
00617 FlightPeriodParser::definition<ScannerT>::start() const {
00618     return flight_period_list;
00619 }
00620 }
00621
00622 //
00623 // Entry class for the file parser
00624 //
00625 // //////////////////////////////////////
00626 FlightPeriodFileParser::
00627 FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00628                         const stdair::Filename_T& iFilename)
00629 : _filename (iFilename), _bomRoot (ioBomRoot) {
00630     init();
00631 }
00632
00633 // //////////////////////////////////////
00634 void FlightPeriodFileParser::init() {
00635     // Open the file
00636     _startIterator = iterator_t (_filename);
00637 }

```

```

00642 // Check the filename exists and can be open
00643 if (!_startIterator) {
00644     std::ostream oMessage;
00645     oMessage << "The file " << _filename << " can not be open." << std::endl;
00646     STDAIR_LOG_ERROR (oMessage.str());
00647     throw ScheduleInputFileNotFoundException (oMessage.str());
00648 }
00649
00650 // Create an EOF iterator
00651 _endIterator = _startIterator.make_end();
00652 }
00653
00654 // //////////////////////////////////////
00655 bool FlightPeriodFileParser::generateInventories () {
00656     bool oResult = false;
00657
00658     STDAIR_LOG_DEBUG ("Parsing schedule input file: " << _filename);
00659
00660     // Initialise the parser (grammar) with the helper/staging structure.
00661     ScheduleParserHelper::FlightPeriodParser lFPParser (_bomRoot, _flightPeriod
);
00662
00663     // Launch the parsing of the file and, thanks to the doEndFlight
00664     // call-back structure, the building of the whole BomRoot BOM
00665     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00666     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
,
00667                                             lFPParser,
00668                                             bsc::space_p - bsc::eol_p);
00669
00670     // Retrieves whether or not the parsing was successful
00671     oResult = info.hit;
00672
00673     const bool isFull = info.full;
00674
00675     const std::string hasBeenFullyReadStr = (isFull == true)?"":"not ";
00676     if (oResult == true && isFull == true) {
00677         STDAIR_LOG_DEBUG ("Parsing of schedule input file: " << _filename
00678                         << " succeeded: read " << info.length
00679                         << " characters. The input file has "
00680                         << hasBeenFullyReadStr
00681                         << "been fully read. Stop point: " << info.stop);
00682     } else {
00683         STDAIR_LOG_ERROR ("Parsing of schedule input file: " << _filename
00684                         << " failed: read " << info.length
00685                         << " characters. The input file has "
00686                         << hasBeenFullyReadStr
00687                         << "been fully read. Stop point: " << info.stop);
00688         throw ScheduleFileParsingFailedException ("Parsing of schedule input
00689 file: "
00690                                             + _filename + " failed.");
00691     }
00692
00693     return oResult;
00694 }
00695
00696 }

```

25.135 airinv/command/ScheduleParserHelper.hpp File Reference

```

#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>

```

Classes

- struct [AIRINV::ScheduleParserHelper::ParserSemanticAction](#)
- struct [AIRINV::ScheduleParserHelper::storeAirlineCode](#)
- struct [AIRINV::ScheduleParserHelper::storeFlightNumber](#)
- struct [AIRINV::ScheduleParserHelper::storeDateRangeStart](#)
- struct [AIRINV::ScheduleParserHelper::storeDateRangeEnd](#)

- struct [AIRINV::ScheduleParserHelper::storeDow](#)
- struct [AIRINV::ScheduleParserHelper::storeLegBoardingPoint](#)
- struct [AIRINV::ScheduleParserHelper::storeLegOffPoint](#)
- struct [AIRINV::ScheduleParserHelper::storeBoardingTime](#)
- struct [AIRINV::ScheduleParserHelper::storeOffTime](#)
- struct [AIRINV::ScheduleParserHelper::storeElapsedTime](#)
- struct [AIRINV::ScheduleParserHelper::storeLegCabinCode](#)
- struct [AIRINV::ScheduleParserHelper::storeCapacity](#)
- struct [AIRINV::ScheduleParserHelper::storeSegmentSpecificity](#)
- struct [AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint](#)
- struct [AIRINV::ScheduleParserHelper::storeSegmentOffPoint](#)
- struct [AIRINV::ScheduleParserHelper::storeSegmentCabinCode](#)
- struct [AIRINV::ScheduleParserHelper::storeClasses](#)
- struct [AIRINV::ScheduleParserHelper::storeFamilyCode](#)
- struct [AIRINV::ScheduleParserHelper::storeFClasses](#)
- struct [AIRINV::ScheduleParserHelper::doEndFlight](#)
- struct [AIRINV::ScheduleParserHelper::FlightPeriodParser](#)
- struct [AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >](#)
- class [AIRINV::FlightPeriodFileParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::ScheduleParserHelper](#)

25.136 ScheduleParserHelper.hpp

```

00001 #ifndef __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP
00002 #define __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FlightPeriodStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022     namespace ScheduleParserHelper {
00023
00024         // //////////////////////////////////////
00025         // Semantic actions
00026         // //////////////////////////////////////
00027
00028         struct ParserSemanticAction {
00029             ParserSemanticAction (FlightPeriodStruct&);
00030             FlightPeriodStruct& _flightPeriod;
00031         };
00032
00033         struct storeAirlineCode : public ParserSemanticAction {
00034             storeAirlineCode (FlightPeriodStruct&);
00035             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00036         };
00037
00038         struct storeFlightNumber : public ParserSemanticAction {
00039             storeFlightNumber (FlightPeriodStruct&);

```



```
00049     void operator() (unsigned int iNumber) const;
00050 };
00051
00053 struct storeDateRangeStart : public ParserSemanticAction {
00055     storeDateRangeStart (FlightPeriodStruct&);
00057     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00058 };
00059
00061 struct storeDateRangeEnd : public ParserSemanticAction {
00063     storeDateRangeEnd (FlightPeriodStruct&);
00065     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00066 };
00067
00069 struct storeDow : public ParserSemanticAction {
00071     storeDow (FlightPeriodStruct&);
00073     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00074 };
00075
00077 struct storeLegBoardingPoint : public ParserSemanticAction {
00079     storeLegBoardingPoint (FlightPeriodStruct&);
00081     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00082 };
00083
00085 struct storeLegOffPoint : public ParserSemanticAction {
00087     storeLegOffPoint (FlightPeriodStruct&);
00089     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00090 };
00091
00093 struct storeBoardingTime : public ParserSemanticAction {
00095     storeBoardingTime (FlightPeriodStruct&);
00097     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00098 };
00099
00101 struct storeOffTime : public ParserSemanticAction {
00103     storeOffTime (FlightPeriodStruct&);
00105     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00106 };
00107
00109 struct storeElapsedTime : public ParserSemanticAction {
00111     storeElapsedTime (FlightPeriodStruct&);
00113     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00114 };
00115
00117 struct storeLegCabinCode : public ParserSemanticAction {
00119     storeLegCabinCode (FlightPeriodStruct&);
00121     void operator() (char iChar) const;
00122 };
00123
00125 struct storeCapacity : public ParserSemanticAction {
00127     storeCapacity (FlightPeriodStruct&);
00129     void operator() (double iReal) const;
00130 };
00131
00136 struct storeSegmentSpecificity : public ParserSemanticAction {
00138     storeSegmentSpecificity (FlightPeriodStruct&);
00140     void operator() (char iChar) const;
00141 };
00142
00144 struct storeSegmentBoardingPoint : public ParserSemanticAction {
00146     storeSegmentBoardingPoint (FlightPeriodStruct&);
00148     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00149 };
00150
00152 struct storeSegmentOffPoint : public ParserSemanticAction {
00154     storeSegmentOffPoint (FlightPeriodStruct&);
00156     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00157 };
00158
00160 struct storeSegmentCabinCode : public ParserSemanticAction {
00162     storeSegmentCabinCode (FlightPeriodStruct&);
00164     void operator() (char iChar) const;
00165 };
00166
00168 struct storeClasses : public ParserSemanticAction {
00170     storeClasses (FlightPeriodStruct&);
00172     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00173 };
00174
00176 struct storeFamilyCode : public ParserSemanticAction {
00178     storeFamilyCode (FlightPeriodStruct&);
00180     void operator() (int iCode) const;
00181 };
00182
00184 struct storeFClasses : public ParserSemanticAction {
00186     storeFClasses (FlightPeriodStruct&);
00188     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00189 };
```

```

00190
00192 struct doEndFlight : public ParserSemanticAction {
00194     doEndFlight (stdair::BomRoot&, FlightPeriodStruct&);
00196     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00198     stdair::BomRoot& _bomRoot;
00199 };
00200
00201
00203 //
00204 // (Boost Spirit) Grammar Definition
00205 //
00207
00249 struct FlightPeriodParser :
00250     public boost::spirit::classic::grammar<FlightPeriodParser> {
00251
00252     FlightPeriodParser (stdair::BomRoot&, FlightPeriodStruct&);
00253
00254     template <typename ScannerT>
00255     struct definition {
00256         definition (FlightPeriodParser const& self);
00257
00258         // Instantiation of rules
00259         boost::spirit::classic::rule<ScannerT> flight_period_list,
00260             not_to_be_parsed, flight_period, flight_period_end,
00261             flight_key, airline_code, flight_number,
00262             date, dow, time, date_offset,
00263             leg, leg_key, leg_details, leg_cabin_details,
00264             segment_section, segment_key, full_segment_cabin_details,
00265             segment_cabin_details, full_family_cabin_details,
00266             family_cabin_details, generic_segment, specific_segment_list;
00267
00269         boost::spirit::classic::rule<ScannerT> const& start() const;
00270     };
00271
00272     // Parser Context
00273     stdair::BomRoot& _bomRoot;
00274     FlightPeriodStruct& _flightPeriod;
00275 };
00276
00277 }
00282
00283 //
00284 // Entry class for the file parser
00285 //
00287
00292 class FlightPeriodFileParser : public stdair::CmdAbstract {
00293 public:
00295     FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00296                             const stdair::Filename_T& iFilename);
00297
00299     bool generateInventories ();
00300
00301 private:
00303     void init();
00304
00305 private:
00306     // Attributes
00308     stdair::Filename_T _filename;
00309
00311     iterator_t _startIterator;
00312
00314     iterator_t _endIterator;
00315
00317     stdair::BomRoot& _bomRoot;
00318
00320     FlightPeriodStruct _flightPeriod;
00321 };
00322
00323 }
00324 #endif // __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP

```

25.137 airinv/command/vault/DCPEventGenerator.cpp File Reference

```

#include <cassert>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/DCPEventStruct.hpp>
#include <airinv/command/DCPEventGenerator.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.138 DCPEventGenerator.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/bom/BomManager.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/factory/FacBomManager.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirInv
00012 #include <airinv/bom/DCPEventStruct.hpp>
00013 #include <airinv/command/DCPEventGenerator.hpp>
00014
00015 namespace AIRINV {
00016
00017 // //////////////////////////////////////
00018 void DCPEventGenerator::
00019     createDCPEvent (stdair::BomRoot& ioBomRoot,
00020                     DCPEventStruct& iDCPEventStruct) {
00021
00022     // Set the airport-pair primary key.
00023     /*
00024     const stdair::AirportCode_T& lBoardPoint = iDCPEventStruct._origin;
00025     const stdair::AirportCode_T& lOffPoint = iDCPEventStruct._destination;
00026     */
00027
00028     // Set the DCP date-period primary key.
00029     const stdair::Date_T& lDateRangeStart = iDCPEventStruct._dateRangeStart;
00030     const stdair::Date_T& lDateRangeEnd = iDCPEventStruct._dateRangeEnd;
00031     const stdair::DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
00032
00033     // Set the DCP time-period primary key.
00034     /*
00035     const stdair::Time_T& lTimeRangeStart = iDCPEventStruct._timeRangeStart;
00036     const stdair::Time_T& lTimeRangeEnd = iDCPEventStruct._timeRangeEnd;
00037     */
00038
00039     // Generate the DCPEvent
00040     const stdair::DayDuration_T& lAdvancePurchase =
00041         iDCPEventStruct._advancePurchase;
00042     const stdair::SaturdayStay_T& lSaturdayStay = iDCPEventStruct._saturdayStay
;
00043     const stdair::ChangeFees_T& lChangeFees = iDCPEventStruct._changeFees;
00044     const stdair::NonRefundable_T& lNonRefundable =
00045         iDCPEventStruct._nonRefundable;
00046     const stdair::DayDuration_T& lMinimumStay = iDCPEventStruct._minimumStay;
00047     const stdair::Fare_T& lDCP = iDCPEventStruct._DCP;
00048
00049     // Generate Segment Features and link them to their DCPEvent
00050     stdair::ClassList_StringList_T::const_iterator lItCurrentClassCodeList =
00051         iDCPEventStruct._classCodeList.begin();
00052
00053     const unsigned int lAirlineListSize = iDCPEventStruct.getAirlineListSize();
00054     const unsigned int lClassCodeListSize =
00055         iDCPEventStruct.getClassCodeListSize();
00056     assert (lAirlineListSize == lClassCodeListSize);
00057
00058     iDCPEventStruct.beginClassCode();
00059     for (iDCPEventStruct.beginAirline();
00060          iDCPEventStruct.hasNotReachedEndAirline();
00061          iDCPEventStruct.iterateAirline()) {
00062         /*
00063         const stdair::AirlineCode_T& lAirlineCode =
00064             iDCPEventStruct.getCurrentAirlineCode();
00065         const std::string& lClassCodeList =
00066             iDCPEventStruct.getCurrentClassCode();
00067             iDCPEventStruct.iterateClassCode();
00068         */
00068     }
00069 }
00070
00071 }

```

00072

25.139 airinv/command/vault/DCPEventGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class [AIRINV::DCPEventGenerator](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::DCPParserHelper](#)

25.140 DCPEventGenerator.hpp

```
00001 #ifndef __AIRINV_CMD_DCPEVENTGENERATOR_HPP
00002 #define __AIRINV_CMD_DCPEVENTGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirInv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     class BomRoot;
00015     class DCPEvent;
00016 }
00017
00018 namespace AIRINV {
00019
00020     // Forward declarations
00021     struct DCPEventStruct;
00022     namespace DCPParserHelper {
00023         struct doEndDCP;
00024     }
00025
00027     class DCPEventGenerator : public stdair::CmdAbstract {
00028         // Only the following class may use methods of DCPGenerator.
00029         // Indeed, as those methods build the BOM, it is not good to expose
00030         // them public.
00031         friend class DCPFileParser;
00032         friend struct DCPParserHelper::doEndDCP;
00033         friend class DCPParser;
00034     private:
00037         static void createDCPEvent (stdair::BomRoot&, DCPEventStruct&);
00038     };
00039
00040 }
00041 #endif // __AIRINV_CMD_DCPEVENTGENERATOR_HPP
```

25.141 airinv/command/vault/DCPParser.cpp File Reference

```
#include <cassert>
#include <string>
#include <stdair/service/Logger.hpp>
#include <airinv/command/DCPParserHelper.hpp>
#include <airinv/command/DCPParser.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.142 DCPParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // AirSched
00010 #include <airinv/command/DCPParserHelper.hpp>
00011 #include <airinv/command/DCPParser.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 void DCPParser::DCPRuleGeneration (const stdair::Filename_T& iFilename,
00017                                     stdair::BomRoot& ioBomRoot) {
00018
00019     // Initialise the DCP file parser
00020     DCPRuleFileParser lDCPRuleFileParser (ioBomRoot, iFilename);
00021
00022     // Parse the CSV-formatted DCP input file and generate the
00023     // corresponding DCP events
00024     lDCPRuleFileParser.generateDCPRules();
00025 }
00026
00027 }

```

25.143 airinv/command/vault/DCPParser.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [AIRINV::DCPParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.144 DCPParser.hpp

```

00001 #ifndef __AIRINV_CMD_DCPPARSER_HPP
00002 #define __AIRINV_CMD_DCPPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00011 // Forward declarations.
00012 namespace stdair {

```

```

00013     class BomRoot;
00014 }
00015
00016 namespace AIRINV {
00017
00019     class DCPParser : public stdair::CmdAbstract {
00020     public:
00028         static void DCPRuleGeneration (const stdair::Filename_T&,
00029                                         stdair::BomRoot&);
00030     };
00031 }
00032 #endif // __AIRINV_CMD_DCPPARSER_HPP

```

25.145 airinv/command/vault/DCPParserHelper.cpp File Reference

```

#include <cassert>
#include <string>
#include <vector>
#include <fstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/DCPParserHelper.hpp>
#include <airinv/command/DCPRuleGenerator.hpp>

```

Namespaces

- namespace [AIRINV](#)
- namespace [AIRINV::DCPParserHelper](#)

Variables

- stdair::int1_p_t [AIRINV::DCPParserHelper::int1_p](#)
- stdair::uint2_p_t [AIRINV::DCPParserHelper::uint2_p](#)
- stdair::uint4_p_t [AIRINV::DCPParserHelper::uint4_p](#)
- stdair::uint1_4_p_t [AIRINV::DCPParserHelper::uint1_4_p](#)
- stdair::hour_p_t [AIRINV::DCPParserHelper::hour_p](#)
- stdair::minute_p_t [AIRINV::DCPParserHelper::minute_p](#)
- stdair::second_p_t [AIRINV::DCPParserHelper::second_p](#)
- stdair::year_p_t [AIRINV::DCPParserHelper::year_p](#)
- stdair::month_p_t [AIRINV::DCPParserHelper::month_p](#)
- stdair::day_p_t [AIRINV::DCPParserHelper::day_p](#)

25.146 DCPParserHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <vector>
00008 #include <fstream>
00009 // StdAir
00010 #include <stdair/basic/BasFileMgr.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/service/Logger.hpp>
00013 // AirInv
00014 #include <airinv/command/DCPParserHelper.hpp>
00015 #include <airinv/command/DCPRuleGenerator.hpp>
00016
00017 namespace AIRINV {
00018

```

```

00019 namespace DCPParserHelper {
00020
00021 // //////////////////////////////////////
00022 // Semantic actions
00023 // //////////////////////////////////////
00024
00025 ParserSemanticAction::ParserSemanticAction (DCPRuleStruct& ioDCPRule)
00026 : _DCPRule (ioDCPRule) {
00027 }
00028
00029 // //////////////////////////////////////
00030 storeDCPId::storeDCPId (DCPRuleStruct& ioDCPRule)
00031 : ParserSemanticAction (ioDCPRule) {
00032 }
00033
00034 // //////////////////////////////////////
00035 void storeDCPId::operator() (unsigned int iDCPId,
00036                             boost::spirit::qi::unused_type,
00037                             boost::spirit::qi::unused_type) const {
00038     _DCPRule._DCPId = iDCPId;
00039
00040     // DEBUG
00041     //STDAIR_LOG_DEBUG ( "DCP Id: " << _DCPRule._DCPId);
00042
00043     _DCPRule._nbOfAirlines = 0;
00044     _DCPRule._airlineCode = "";
00045     _DCPRule._classCode = "";
00046     _DCPRule._airlineCodeList.clear();
00047     _DCPRule._classCodeList.clear();
00048     _DCPRule._classCodeListOfList.clear();
00049     _DCPRule._itSeconds = 0;
00050 }
00051
00052 // //////////////////////////////////////
00053 storeOrigin ::
00054 storeOrigin (DCPRuleStruct& ioDCPRule)
00055 : ParserSemanticAction (ioDCPRule) {
00056 }
00057
00058 // //////////////////////////////////////
00059 void storeOrigin::operator() (std::vector<char> iChar,
00060                             boost::spirit::qi::unused_type,
00061                             boost::spirit::qi::unused_type) const {
00062     stdair::AirportCode_T lOrigin (iChar.begin(), iChar.end());
00063     // DEBUG
00064     //STDAIR_LOG_DEBUG ( "Origin: " << lOrigin);
00065     _DCPRule._origin = lOrigin;
00066 }
00067
00068 // //////////////////////////////////////
00069 storeDestination ::
00070 storeDestination (DCPRuleStruct& ioDCPRule)
00071 : ParserSemanticAction (ioDCPRule) {
00072 }
00073
00074 // //////////////////////////////////////
00075 void storeDestination::operator() (std::vector<char> iChar,
00076                                   boost::spirit::qi::unused_type,
00077                                   boost::spirit::qi::unused_type) const {
00078     stdair::AirportCode_T lDestination (iChar.begin(), iChar.end());
00079     // DEBUG
00080     //STDAIR_LOG_DEBUG ( "Destination: " << lDestination);
00081     _DCPRule._destination = lDestination;
00082 }
00083
00084 // //////////////////////////////////////
00085 storeDateRangeStart::
00086 storeDateRangeStart (DCPRuleStruct& ioDCPRule)
00087 : ParserSemanticAction (ioDCPRule) {
00088 }
00089
00090 // //////////////////////////////////////
00091 void storeDateRangeStart::operator() (boost::spirit::qi::unused_type,
00092                                       boost::spirit::qi::unused_type,
00093                                       boost::spirit::qi::unused_type) const
00094 {
00095     _DCPRule._dateRangeStart = _DCPRule.getDate();
00096     // DEBUG
00097     //STDAIR_LOG_DEBUG ( "Date Range Start: " << _DCPRule._dateRangeStart);
00098 }
00099
00100 // //////////////////////////////////////
00101 storeDateRangeEnd::
00102 storeDateRangeEnd (DCPRuleStruct& ioDCPRule)
00103 : ParserSemanticAction (ioDCPRule) {
00104 }

```

```

00105 // //////////////////////////////////////
00106 void storeDateRangeEnd::operator() (boost::spirit::qi::unused_type,
00107                                     boost::spirit::qi::unused_type,
00108                                     boost::spirit::qi::unused_type) const {
00109     _DCPRule._dateRangeEnd = _DCPRule.getDate();
00110     // DEBUG
00111     //STDAIR_LOG_DEBUG ("Date Range End: " << _DCPRule._dateRangeEnd);
00112 }
00113 // //////////////////////////////////////
00114 storeStartTime::
00115 storeStartTime (DCPRuleStruct& ioDCPRule)
00116 : ParserSemanticAction (ioDCPRule) {
00117 }
00118 // //////////////////////////////////////
00121 void storeStartTime::operator() (boost::spirit::qi::unused_type,
00122                                 boost::spirit::qi::unused_type,
00123                                 boost::spirit::qi::unused_type) const
00124 {
00125     _DCPRule._timeRangeStart = _DCPRule.getTime();
00126     // DEBUG
00127     //STDAIR_LOG_DEBUG ("Time Range Start: " << _DCPRule._timeRangeStart);
00128     // Reset the number of seconds
00129     _DCPRule._itSeconds = 0;
00130 }
00131 // //////////////////////////////////////
00132 storeEndTime::
00133 storeEndTime (DCPRuleStruct& ioDCPRule)
00134 : ParserSemanticAction (ioDCPRule) {
00135 }
00136 // //////////////////////////////////////
00138 void storeEndTime::operator() (boost::spirit::qi::unused_type,
00139                                 boost::spirit::qi::unused_type,
00140                                 boost::spirit::qi::unused_type) const {
00141     _DCPRule._timeRangeEnd = _DCPRule.getTime();
00142     // DEBUG
00143     //STDAIR_LOG_DEBUG ("Time Range End: " << _DCPRule._timeRangeEnd);
00144     // Reset the number of seconds
00145     _DCPRule._itSeconds = 0;
00146 }
00147 // //////////////////////////////////////
00148 storePOS ::
00149 storePOS (DCPRuleStruct& ioDCPRule)
00150 : ParserSemanticAction (ioDCPRule) {
00151 }
00152 // //////////////////////////////////////
00155 void storePOS::operator() (std::vector<char> iChar,
00156                             boost::spirit::qi::unused_type,
00157                             boost::spirit::qi::unused_type) const {
00158     stdair::AirlineCode_T lPOS (iChar.begin(), iChar.end());
00159     _DCPRule._pos = lPOS;
00160     // DEBUG
00161     //STDAIR_LOG_DEBUG ("POS: " << _DCPRule._pos);
00162 }
00163 // //////////////////////////////////////
00164 storeCabinCode ::
00165 storeCabinCode (DCPRuleStruct& ioDCPRule)
00166 : ParserSemanticAction (ioDCPRule) {
00167 }
00168 // //////////////////////////////////////
00171 void storeCabinCode::operator() (char iChar,
00172                                 boost::spirit::qi::unused_type,
00173                                 boost::spirit::qi::unused_type) const {
00174     std::ostringstream ostr;
00175     ostr << iChar;
00176     std::string cabinCodeStr = ostr.str();
00177     const stdair::CabinCode_T lCabinCode (cabinCodeStr);
00178     _DCPRule._cabinCode = lCabinCode;
00179     // DEBUG
00180     //STDAIR_LOG_DEBUG ("Cabin Code: " << lCabinCode);
00181 }
00182 // //////////////////////////////////////
00183 storeChannel ::
00184 storeChannel (DCPRuleStruct& ioDCPRule)
00185 : ParserSemanticAction (ioDCPRule) {
00186 }
00187 }
00188 }
00189 }
00190

```



```

00191 // //////////////////////////////////////
00192 void storeChannel::operator() (std::vector<char> iChar,
00193                               boost::spirit::qi::unused_type,
00194                               boost::spirit::qi::unused_type) const {
00195     stdair::ChannelLabel_T lChannel (iChar.begin(), iChar.end());
00196     if (lChannel != "IN" && lChannel != "IF"
00197         && lChannel != "DN" && lChannel != "DF") {
00198         // DEBUG
00199         STDAIR_LOG_DEBUG ("Invalid channel " << lChannel);
00200     }
00201     _DCPRule._channel = lChannel;
00202     // DEBUG
00203     //STDAIR_LOG_DEBUG ("Channel: " << _DCPRule._channel);
00204 }
00205
00206 // //////////////////////////////////////
00207 storeAdvancePurchase ::
00208 storeAdvancePurchase (DCPRuleStruct& ioDCPRule)
00209 : ParserSemanticAction (ioDCPRule) {
00210 }
00211
00212 // //////////////////////////////////////
00213 void storeAdvancePurchase::operator() (unsigned int iAdvancePurchase,
00214                                       boost::spirit::qi::unused_type,
00215                                       boost::spirit::qi::unused_type)
00216 const {
00217     _DCPRule._advancePurchase = iAdvancePurchase;
00218     // DEBUG
00219     //STDAIR_LOG_DEBUG ( "Advance Purchase: " << _DCPRule._advancePurchase);
00220 }
00221
00222 // //////////////////////////////////////
00223 storeSaturdayStay ::
00224 storeSaturdayStay (DCPRuleStruct& ioDCPRule)
00225 : ParserSemanticAction (ioDCPRule) {
00226 }
00227
00228 // //////////////////////////////////////
00229 void storeSaturdayStay::operator() (char iSaturdayStay,
00230                                    boost::spirit::qi::unused_type,
00231                                    boost::spirit::qi::unused_type) const {
00232     bool lBool = false;
00233     if (iSaturdayStay == 'T') {
00234         lBool = true;
00235     } else {
00236         if (iSaturdayStay != 'F') {
00237             // DEBUG
00238             STDAIR_LOG_DEBUG ("Invalid saturdayStay char " << iSaturdayStay);
00239         }
00240     }
00241     stdair::SaturdayStay_T lSaturdayStay (lBool);
00242     _DCPRule._saturdayStay = lSaturdayStay;
00243     // DEBUG
00244     //STDAIR_LOG_DEBUG ("Saturday Stay: " << _DCPRule._saturdayStay);
00245 }
00246
00247 // //////////////////////////////////////
00248 storeChangeFees ::
00249 storeChangeFees (DCPRuleStruct& ioDCPRule)
00250 : ParserSemanticAction (ioDCPRule) {
00251 }
00252
00253 // //////////////////////////////////////
00254 void storeChangeFees::operator() (char iChangefees,
00255                                   boost::spirit::qi::unused_type,
00256                                   boost::spirit::qi::unused_type) const {
00257     bool lBool = false;
00258     if (iChangefees == 'T') {
00259         lBool = true;
00260     } else {
00261         if (iChangefees != 'F') {
00262             // DEBUG
00263             STDAIR_LOG_DEBUG ("Invalid change fees char " << iChangefees);
00264         }
00265     }
00266     stdair::ChangeFees_T lChangefees (lBool);
00267     _DCPRule._changeFees = lChangefees;
00268     // DEBUG
00269     //STDAIR_LOG_DEBUG ("Change fees: " << _DCPRule._changeFees);
00270 }
00271
00272 // //////////////////////////////////////
00273 storeNonRefundable ::
00274 storeNonRefundable (DCPRuleStruct& ioDCPRule)
00275 : ParserSemanticAction (ioDCPRule) {
00276 }

```

```

00277
00278 ///////////////////////////////////////////////////////////////////
00279 void storeNonRefundable::operator() (char iNonRefundable,
00280                                     boost::spirit::qi::unused_type,
00281                                     boost::spirit::qi::unused_type) const
00282 {
00283     bool lBool = false;
00284     if (iNonRefundable == 'T') {
00285         lBool = true;
00286     } else {
00287         if (iNonRefundable != 'F') {
00288             // DEBUG
00289             STDAIR_LOG_DEBUG ("Invalid non refundable char " << iNonRefundable);
00290         }
00291         stdair::NonRefundable_T lNonRefundable (lBool);
00292         _DCPRule._nonRefundable = lNonRefundable;
00293         // DEBUG
00294         //STDAIR_LOG_DEBUG ("Non refundable: " << _DCPRule._nonRefundable);
00295     }
00296
00297 ///////////////////////////////////////////////////////////////////
00298 storeMinimumStay ::
00299 storeMinimumStay (DCPRuleStruct& ioDCPRule)
00300 : ParserSemanticAction (ioDCPRule) {
00301 }
00302
00303 ///////////////////////////////////////////////////////////////////
00304 void storeMinimumStay::operator() (unsigned int iMinStay,
00305                                   boost::spirit::qi::unused_type,
00306                                   boost::spirit::qi::unused_type) const {
00307     _DCPRule._minimumStay = iMinStay;
00308     // DEBUG
00309     //STDAIR_LOG_DEBUG ("Minimum Stay: " << _DCPRule._minimumStay );
00310 }
00311
00312 ///////////////////////////////////////////////////////////////////
00313 storeDCP ::
00314 storeDCP (DCPRuleStruct& ioDCPRule)
00315 : ParserSemanticAction (ioDCPRule) {
00316 }
00317
00318 ///////////////////////////////////////////////////////////////////
00319 void storeDCP::operator() (double iDCP,
00320                           boost::spirit::qi::unused_type,
00321                           boost::spirit::qi::unused_type) const {
00322     _DCPRule._DCP = iDCP;
00323     // DEBUG
00324     //STDAIR_LOG_DEBUG ("DCP: " << _DCPRule._DCP);
00325 }
00326
00327 ///////////////////////////////////////////////////////////////////
00328 storeAirlineCode ::
00329 storeAirlineCode (DCPRuleStruct& ioDCPRule)
00330 : ParserSemanticAction (ioDCPRule) {
00331 }
00332
00333 ///////////////////////////////////////////////////////////////////
00334 void storeAirlineCode::operator() (std::vector<char> iChar,
00335                                   boost::spirit::qi::unused_type,
00336                                   boost::spirit::qi::unused_type) const {
00337     bool lAlreadyInTheList = false;
00338     stdair::AirlineCode_T lAirlineCode (iChar.begin(), iChar.end());
00339     // Update the airline code
00340     _DCPRule._airlineCode = lAirlineCode;
00341     // Test if the DCPRule Struct stands for interline products
00342     if (_DCPRule._airlineCodeList.size() > 0) {
00343         _DCPRule._classCodeListOfList.push_back(_DCPRule._classCodeList);
00344         _DCPRule._classCodeList.clear();
00345         // Update the number of airlines if necessary
00346         std::vector<stdair::AirlineCode_T>::iterator Airline_iterator;
00347         for (Airline_iterator = _DCPRule._airlineCodeList.begin();
00348             Airline_iterator != _DCPRule._airlineCodeList.end();
00349             ++Airline_iterator) {
00350             stdair::AirlineCode_T lPreviousAirlineCode =
00351                 *Airline_iterator;
00352             if (lPreviousAirlineCode == lAirlineCode) {
00353                 lAlreadyInTheList = true;
00354                 /*STDAIR_LOG_DEBUG ("Airline Code Already Existing: "
00355                                     << lAirlineCode);*/
00356             }
00357         }
00358     }
00359     if (lAlreadyInTheList == false) {
00360         /*STDAIR_LOG_DEBUG ("New Airline Code: "
00361                             << lAirlineCode);*/
00362         _DCPRule._airlineCodeList.push_back(lAirlineCode);

```

```

00363         _DCPRule._classCodeList.clear();
00364     }
00365     } else {
00366         /*STDAIR_LOG_DEBUG ("First Airline Code: "
00367             << lAirlineCode);*/
00368         _DCPRule._airlineCodeList.push_back (lAirlineCode);
00369     }
00370     // DEBUG
00371     //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode);
00372 }
00373
00374 ///////////////////////////////////////////////////////////////////
00375 storeClass ::
00376 storeClass (DCPRuleStruct& ioDCPRule)
00377     : ParserSemanticAction (ioDCPRule) {
00378 }
00379
00380 ///////////////////////////////////////////////////////////////////
00381 void storeClass::operator() (std::vector<char> iChar,
00382     boost::spirit::qi::unused_type,
00383     boost::spirit::qi::unused_type) const {
00384     std::ostringstream ostr;
00385     for (std::vector<char>::const_iterator lItVector = iChar.begin();
00386         lItVector != iChar.end();
00387         lItVector++) {
00388         ostr << *lItVector;
00389     }
00390     std::string classCodeStr = ostr.str();
00391     // Insertion of this class Code list in the whole classCode name
00392     _DCPRule._classCodeList.push_back(classCodeStr);
00393     // DEBUG
00394     // STDAIR_LOG_DEBUG ("Class Code: " << classCodeStr);
00395 }
00396
00397 ///////////////////////////////////////////////////////////////////
00398 doEndDCP::
00399 doEndDCP (stdair::BomRoot& ioBomRoot,
00400     DCPRuleStruct& ioDCPRule)
00401     : ParserSemanticAction (ioDCPRule),
00402     _bomRoot (ioBomRoot) {
00403 }
00404
00405 ///////////////////////////////////////////////////////////////////
00406 void doEndDCP::operator() (boost::spirit::qi::unused_type,
00407     boost::spirit::qi::unused_type,
00408     boost::spirit::qi::unused_type) const {
00409     // DEBUG
00410     // STDAIR_LOG_DEBUG ("Do End");
00411     // Generation of the DCP rule object.
00412     _DCPRule._classCodeListOfList.push_back(_DCPRule._classCodeList);
00413     DCPRuleGenerator::createDCPRule (_bomRoot, _DCPRule);
00414     STDAIR_LOG_DEBUG(_DCPRule.describe());
00415 }
00416
00417 ///////////////////////////////////////////////////////////////////
00418 //
00419 // Utility Parsers
00420 //
00421 ///////////////////////////////////////////////////////////////////
00422 namespace bsq = boost::spirit::qi;
00423 namespace bsa = boost::spirit::ascii;
00424
00425 stdair::int1_p_t int1_p;
00426
00427 stdair::uint2_p_t uint2_p;
00428
00429 stdair::uint4_p_t uint4_p;
00430
00431 stdair::uint1_4_p_t uint1_4_p;
00432
00433 stdair::hour_p_t hour_p;
00434 stdair::minute_p_t minute_p;
00435 stdair::second_p_t second_p;
00436
00437 stdair::year_p_t year_p;
00438 stdair::month_p_t month_p;
00439 stdair::day_p_t day_p;
00440
00441 ///////////////////////////////////////////////////////////////////
00442 // (Boost Spirit) Grammar Definition
00443 ///////////////////////////////////////////////////////////////////
00444
00445 ///////////////////////////////////////////////////////////////////
00446 DCPRuleParser::DCPRuleParser (stdair::BomRoot& ioBomRoot,
00447     DCPRuleStruct& ioDCPRule) :
00448     DCPRuleParser::base_type(start),
00449     _bomRoot(ioBomRoot), _DCPRule(ioDCPRule) {

```

```

00457
00458     start = *(comments | DCP_rule);
00459
00460     comments = (bsq::lexeme[bsq::repeat(2) [bsa::char_('/')]]
00461                 >> +(bsa::char_ - bsq::eol)
00462                 >> bsq::eol]
00463                 | bsq::lexeme[bsa::char_('/') >> bsa::char_('*')
00464                 >> +(bsa::char_ - bsa::char_('*'))
00465                 >> bsa::char_('*') >> bsa::char_('/')]);
00466
00467     DCP_rule = DCP_key
00468     >> +(';' >> segment )
00469     >> DCP_rule_end[doEndDCP(_bomRoot, _DCPRule)];
00470
00471     DCP_rule_end = bsa::char_(';');
00472
00473     DCP_key = DCP_id
00474     >> ';' >> origin >> ';' >> destination
00475     >> ';' >> dateRangeStart >> ';' >> dateRangeEnd
00476     >> ';' >> timeRangeStart >> ';' >> timeRangeEnd
00477     >> ';' >> position >> ';' >> cabinCode >> ';' >> channel
00478     >> ';' >> advancePurchase >> ';' >> saturdayStay
00479     >> ';' >> changeFees >> ';' >> nonRefundable
00480     >> ';' >> minimumStay >> ';' >> DCP;
00481
00482     DCP_id = uint1_4_p[storeDCPId(_DCPRule)];
00483
00484     origin = bsq::repeat(3) [bsa::char_("A-Z")] [storeOrigin(_DCPRule)];
00485
00486     destination =
00487     bsq::repeat(3) [bsa::char_("A-Z")] [storeDestination(_DCPRule)];
00488
00489     dateRangeStart = date[storeDateRangeStart(_DCPRule)];
00490
00491     dateRangeEnd = date[storeDateRangeEnd(_DCPRule)];
00492
00493     date = bsq::lexeme
00494     [year_p[boost::phoenix::ref(_DCPRule._itYear) = bsq::labels::_1]
00495     >> '-'
00496     >> month_p[boost::phoenix::ref(_DCPRule._itMonth) = bsq::labels::_1]
00497     >> '-'
00498     >> day_p[boost::phoenix::ref(_DCPRule._itDay) = bsq::labels::_1] ];
00499
00500     timeRangeStart = time[storeStartRangeTime(_DCPRule)];
00501
00502     timeRangeEnd = time[storeEndRangeTime(_DCPRule)];
00503
00504     time = bsq::lexeme
00505     [hour_p[boost::phoenix::ref(_DCPRule._itHours) = bsq::labels::_1]
00506     >> ':'
00507     >> minute_p[boost::phoenix::ref(_DCPRule._itMinutes) = bsq::labels::_1]
00508     >> - (';' >> second_p[boost::phoenix::ref(_DCPRule._itSeconds) =
bsq::labels::_1] ) ];
00509
00510     position = bsq::repeat(3) [bsa::char_("A-Z")] [storePOS(_DCPRule)];
00511
00512     cabinCode = bsa::char_("A-Z") [storeCabinCode(_DCPRule)];
00513
00514     channel = bsq::repeat(2) [bsa::char_("A-Z")] [storeChannel(_DCPRule)];
00515
00516     advancePurchase = uint1_4_p[storeAdvancePurchase(_DCPRule)];
00517
00518     saturdayStay = bsa::char_("A-Z") [storeSaturdayStay(_DCPRule)];
00519
00520     changeFees = bsa::char_("A-Z") [storeChangeFees(_DCPRule)];
00521
00522     nonRefundable = bsa::char_("A-Z") [storeNonRefundable(_DCPRule)];
00523
00524     minimumStay = uint1_4_p[storeMinimumStay(_DCPRule)];
00525
00526     DCP = bsq::double_[storeDCP(_DCPRule)];
00527
00528     segment = bsq::repeat(2) [bsa::char_("A-Z")] [storeAirlineCode(_DCPRule)]
00529     //>> ';'
00530     //>> bsa::char_("A-Z") [storeClass(_DCPRule)]
00531     >> +(';' >> list_class);
00532
00533     list_class = bsq::repeat(1, bsq::inf) [bsa::char_("A-Z")] [storeClass(
_DCPRule)];
00534
00535     //BOOST_SPIRIT_DEBUG_NODE (DCPRuleParser);
00536     BOOST_SPIRIT_DEBUG_NODE (start);
00537     BOOST_SPIRIT_DEBUG_NODE (comments);
00538     BOOST_SPIRIT_DEBUG_NODE (DCP_rule);
00539     BOOST_SPIRIT_DEBUG_NODE (DCP_rule_end);
00540     BOOST_SPIRIT_DEBUG_NODE (DCP_key);

```

```

00541     BOOST_SPIRIT_DEBUG_NODE (DCP_id);
00542     BOOST_SPIRIT_DEBUG_NODE (origin);
00543     BOOST_SPIRIT_DEBUG_NODE (destination);
00544     BOOST_SPIRIT_DEBUG_NODE (dateRangeStart);
00545     BOOST_SPIRIT_DEBUG_NODE (dateRangeEnd);
00546     BOOST_SPIRIT_DEBUG_NODE (date);
00547     BOOST_SPIRIT_DEBUG_NODE (timeRangeStart);
00548     BOOST_SPIRIT_DEBUG_NODE (timeRangeEnd);
00549     BOOST_SPIRIT_DEBUG_NODE (time);
00550     BOOST_SPIRIT_DEBUG_NODE (position);
00551     BOOST_SPIRIT_DEBUG_NODE (cabinCode);
00552     BOOST_SPIRIT_DEBUG_NODE (channel);
00553     BOOST_SPIRIT_DEBUG_NODE (advancePurchase);
00554     BOOST_SPIRIT_DEBUG_NODE (saturdayStay);
00555     BOOST_SPIRIT_DEBUG_NODE (changeFees);
00556     BOOST_SPIRIT_DEBUG_NODE (nonRefundable);
00557     BOOST_SPIRIT_DEBUG_NODE (minimumStay);
00558     BOOST_SPIRIT_DEBUG_NODE (DCP);
00559     BOOST_SPIRIT_DEBUG_NODE (segment);
00560     BOOST_SPIRIT_DEBUG_NODE (list_class);
00561 }
00562 }
00563
00564 //
00565 // Entry class for the file parser
00566 //
00567 //
00568 //
00569 // //////////////////////////////////////
00570 DCPRuleFileParser::
00571 DCPRuleFileParser (stdair::BomRoot& ioBomRoot,
00572                   const stdair::Filename_T& iFilename)
00573 : _filename (iFilename), _bomRoot (ioBomRoot) {
00574     init();
00575 }
00576
00577 // //////////////////////////////////////
00578 void DCPRuleFileParser::init() {
00579     // Check that the file exists and is readable
00580     const bool doesExistAndIsReadable =
00581         stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00582
00583     if (doesExistAndIsReadable == false) {
00584         STDAIR_LOG_ERROR ("The DCP schedule file " << _filename
00585             << " does not exist or can not be read.");
00586
00587         throw DCPInputFileNotFoundException ("The DCP file " + _filename + " does
00588 not exist or can not be read");
00589     }
00590 }
00591
00592 // //////////////////////////////////////
00593 bool DCPRuleFileParser::generatedDCPRules () {
00594     STDAIR_LOG_DEBUG ("Parsing DCP input file: " << _filename);
00595
00596     // File to be parsed
00597     const std::string* lFileName = &_filename;
00598     const char *lChar = (*lFileName).c_str();
00599     std::ifstream fileToBeParsed(lChar, std::ios_base::in);
00600
00601     // Check the filename exists and can be open
00602     if (fileToBeParsed == false) {
00603         STDAIR_LOG_ERROR ("The DCP file " << _filename << " can not be open."
00604             << std::endl);
00605
00606         throw DCPInputFileNotFoundException ("The file " + _filename + " does not
00607 exist or can not be read");
00608     }
00609
00610     // Create an input iterator
00611     stdair::base_iterator_t inputBegin (fileToBeParsed);
00612
00613     // Convert input iterator to an iterator usable by spirit parser
00614     stdair::iterator_t
00615         start (boost::spirit::make_default_multi_pass (inputBegin));
00616     stdair::iterator_t end;
00617
00618     // Initialise the parser (grammar) with the helper/staging structure.
00619     DCPParserHelper::DCPRuleParser lFPParser(_bomRoot, _DCPRule);
00620
00621     // Launch the parsing of the file and, thanks to the doEndDCP
00622     // call-back structure, the building of the whole BomRoot BOM
00623
00624     const bool hasParsingBeenSuccessful =
00625         boost::spirit::qi::phrase_parse (start, end, lFPParser,
00626             boost::spirit::ascii::space);
00627

```

```

00628     if (hasParsingBeenSuccessful == false) {
00629         // TODO: decide whether to throw an exception
00630         STDAIR_LOG_ERROR ("Parsing of DCP input file: " << _filename
00631             << " failed");
00632     }
00633     if (start != end) {
00634         // TODO: decide whether to throw an exception
00635         STDAIR_LOG_ERROR ("Parsing of DCP input file: " << _filename
00636             << " failed");
00637     }
00638     if (hasParsingBeenSuccessful == true && start == end) {
00639         STDAIR_LOG_DEBUG ("Parsing of DCP input file: " << _filename
00640             << " succeeded");
00641     }
00642     return hasParsingBeenSuccessful;
00643 }
00644
00645 }

```

25.147 airinv/command/vault/DCPParserHelper.hpp File Reference

```

#include <stdair/basic/BasParserTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/DCPRuleStruct.hpp>

```

Classes

- struct [AIRINV::DCPParserHelper::ParserSemanticAction](#)
- struct [AIRINV::DCPParserHelper::storeDCPid](#)
- struct [AIRINV::DCPParserHelper::storeOrigin](#)
- struct [AIRINV::DCPParserHelper::storeDestination](#)
- struct [AIRINV::DCPParserHelper::storeDateRangeStart](#)
- struct [AIRINV::DCPParserHelper::storeDateRangeEnd](#)
- struct [AIRINV::DCPParserHelper::storeStartRangeTime](#)
- struct [AIRINV::DCPParserHelper::storeEndRangeTime](#)
- struct [AIRINV::DCPParserHelper::storePOS](#)
- struct [AIRINV::DCPParserHelper::storeCabinCode](#)
- struct [AIRINV::DCPParserHelper::storeChannel](#)
- struct [AIRINV::DCPParserHelper::storeAdvancePurchase](#)
- struct [AIRINV::DCPParserHelper::storeSaturdayStay](#)
- struct [AIRINV::DCPParserHelper::storeChangeFees](#)
- struct [AIRINV::DCPParserHelper::storeNonRefundable](#)
- struct [AIRINV::DCPParserHelper::storeMinimumStay](#)
- struct [AIRINV::DCPParserHelper::storeDCP](#)
- struct [AIRINV::DCPParserHelper::storeAirlineCode](#)
- struct [AIRINV::DCPParserHelper::storeClass](#)
- struct [AIRINV::DCPParserHelper::doEndDCP](#)
- struct [AIRINV::DCPParserHelper::DCPRuleParser](#)
- class [AIRINV::DCPRuleFileParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)
- namespace [AIRINV::DCPParserHelper](#)

25.148 DCPParserHelper.hpp

```

00001 #ifndef __AIRINV_CMD_DCPPARSERHELPER_HPP
00002 #define __AIRINV_CMD_DCPPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 // The stdair/basic/BasParserTypes.hpp header includes Boost.Spirit headers
00009 // #define BOOST_SPIRIT_DEBUG
00010 #include <stdair/basic/BasParserTypes.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012 // AirInv
00013 #include <airinv/AIRINV_Types.hpp>
00014 #include <airinv/bom/DCPRuleStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022     namespace DCPParserHelper {
00023
00024         // //////////////////////////////////////
00025         // Semantic actions
00026         // //////////////////////////////////////
00027
00028         struct ParserSemanticAction {
00029             ParserSemanticAction (DCPRuleStruct&);
00030             DCPRuleStruct& _DCPRule;
00031         };
00032
00033         struct storeDCPId : public ParserSemanticAction {
00034             storeDCPId (DCPRuleStruct&);
00035             void operator() (unsigned int,
00036                             boost::spirit::qi::unused_type,
00037                             boost::spirit::qi::unused_type) const;
00038         };
00039
00040         struct storeOrigin : public ParserSemanticAction {
00041             storeOrigin (DCPRuleStruct&);
00042             void operator() (std::vector<char>,
00043                             boost::spirit::qi::unused_type,
00044                             boost::spirit::qi::unused_type) const;
00045         };
00046
00047         struct storeDestination : public ParserSemanticAction {
00048             storeDestination (DCPRuleStruct&);
00049             void operator() (std::vector<char>,
00050                             boost::spirit::qi::unused_type,
00051                             boost::spirit::qi::unused_type) const;
00052         };
00053
00054         struct storeDateRangeStart : public ParserSemanticAction {
00055             storeDateRangeStart (DCPRuleStruct&);
00056             void operator() (boost::spirit::qi::unused_type,
00057                             boost::spirit::qi::unused_type,
00058                             boost::spirit::qi::unused_type) const;
00059         };
00060
00061         struct storeDateRangeEnd : public ParserSemanticAction {
00062             storeDateRangeEnd (DCPRuleStruct&);
00063             void operator() (boost::spirit::qi::unused_type,
00064                             boost::spirit::qi::unused_type,
00065                             boost::spirit::qi::unused_type) const;
00066         };
00067
00068         struct storeStartRangeTime : public ParserSemanticAction {
00069             storeStartRangeTime (DCPRuleStruct&);
00070             void operator() (boost::spirit::qi::unused_type,
00071                             boost::spirit::qi::unused_type,
00072                             boost::spirit::qi::unused_type) const;
00073         };
00074
00075         struct storeEndRangeTime : public ParserSemanticAction {
00076             storeEndRangeTime (DCPRuleStruct&);
00077             void operator() (boost::spirit::qi::unused_type,
00078                             boost::spirit::qi::unused_type,
00079                             boost::spirit::qi::unused_type) const;
00080         };
00081
00082         struct storePOS : public ParserSemanticAction {
00083             storePOS (DCPRuleStruct&);
00084             void operator() (std::vector<char>,
00085

```

```

00113         boost::spirit::qi::unused_type,
00114         boost::spirit::qi::unused_type) const;
00115     };
00116
00118     struct storeCabinCode : public ParserSemanticAction {
00120         storeCabinCode (DCPRuleStruct&);
00122         void operator() (char,
00123             boost::spirit::qi::unused_type,
00124             boost::spirit::qi::unused_type) const;
00125     };
00126
00128     struct storeChannel : public ParserSemanticAction {
00130         storeChannel (DCPRuleStruct&);
00132         void operator() (std::vector<char>,
00133             boost::spirit::qi::unused_type,
00134             boost::spirit::qi::unused_type) const;
00135     };
00136
00138     struct storeAdvancePurchase : public ParserSemanticAction {
00140         storeAdvancePurchase (DCPRuleStruct&);
00142         void operator() (unsigned int,
00143             boost::spirit::qi::unused_type,
00144             boost::spirit::qi::unused_type) const;
00145     };
00146
00148     struct storeSaturdayStay : public ParserSemanticAction {
00150         storeSaturdayStay (DCPRuleStruct&);
00152         void operator() (char,
00153             boost::spirit::qi::unused_type,
00154             boost::spirit::qi::unused_type) const;
00155     };
00156
00158     struct storeChangeFees : public ParserSemanticAction {
00160         storeChangeFees (DCPRuleStruct&);
00162         void operator() (char,
00163             boost::spirit::qi::unused_type,
00164             boost::spirit::qi::unused_type) const;
00165     };
00166
00168     struct storeNonRefundable : public ParserSemanticAction {
00170         storeNonRefundable (DCPRuleStruct&);
00172         void operator() (char,
00173             boost::spirit::qi::unused_type,
00174             boost::spirit::qi::unused_type) const;
00175     };
00176
00178     struct storeMinimumStay : public ParserSemanticAction {
00180         storeMinimumStay (DCPRuleStruct&);
00182         void operator() (unsigned int,
00183             boost::spirit::qi::unused_type,
00184             boost::spirit::qi::unused_type) const;
00185     };
00186
00188     struct storeDCP : public ParserSemanticAction {
00190         storeDCP (DCPRuleStruct&);
00192         void operator() (double,
00193             boost::spirit::qi::unused_type,
00194             boost::spirit::qi::unused_type) const;
00195     };
00196
00198     struct storeAirlineCode : public ParserSemanticAction {
00200         storeAirlineCode (DCPRuleStruct&);
00202         void operator() (std::vector<char>,
00203             boost::spirit::qi::unused_type,
00204             boost::spirit::qi::unused_type) const;
00205     };
00206
00208     struct storeClass : public ParserSemanticAction {
00210         storeClass (DCPRuleStruct&);
00212         void operator() (std::vector<char>,
00213             boost::spirit::qi::unused_type,
00214             boost::spirit::qi::unused_type) const;
00215     };
00216
00218     struct doEndDCP : public ParserSemanticAction {
00220         doEndDCP (stdair::BomRoot&, DCPRuleStruct&);
00222         void operator() (boost::spirit::qi::unused_type,
00223             boost::spirit::qi::unused_type,
00224             boost::spirit::qi::unused_type) const;
00226         stdair::BomRoot& _bomRoot;
00227     };
00228
00229
00231     //
00232     // (Boost Spirit) Grammar Definition
00233     //
00235

```



```

00304     struct DCPRuleParser :
00305     public boost::spirit::qi::grammar<stdair::iterator_t,
00306                                     boost::spirit::ascii::space_type> {
00307
00308         DCPRuleParser (stdair::BomRoot&, DCPRuleStruct&);
00309
00310         // Instantiation of rules
00311         boost::spirit::qi::rule<stdair::iterator_t,
00312                                 boost::spirit::ascii::space_type>
00313         start, comments, DCP_rule, DCP_rule_end, DCP_key, DCP_id, origin,
00314             destination, dateRangeStart, dateRangeEnd, date, timeRangeStart,
00315             timeRangeEnd, time, position, cabinCode, channel, advancePurchase,
00316             saturdayStay, changeFees, nonRefundable, minimumStay, DCP,
00317             segment, list_class;
00318
00319         // Parser Context
00320         stdair::BomRoot& _bomRoot;
00321         DCPRuleStruct& _DCPRule;
00322     };
00323
00324 }
00325
00326 //
00327 // Entry class for the file parser
00328 //
00329 //
00330
00331 class DCPRuleFileParser : public stdair::CmdAbstract {
00332 public:
00333     DCPRuleFileParser (stdair::BomRoot& ioBomRoot,
00334                       const stdair::Filename_T& iFilename);
00335
00336     bool generateDCPRules ();
00337
00338 private:
00339     void init();
00340
00341 private:
00342     // Attributes
00343     stdair::Filename_T _filename;
00344
00345     stdair::BomRoot& _bomRoot;
00346
00347     DCPRuleStruct _DCPRule;
00348 };
00349
00350 #endif // __AIRINV_CMD_DCPPARSERHELPER_HPP

```

25.149 airinv/config/airinv-paths.hpp File Reference

Defines

- #define PACKAGE "airinv"
- #define PACKAGE_NAME "AIRINV"
- #define PACKAGE_VERSION "0.1.2"
- #define PREFIXDIR "/usr"
- #define EXEC_PREFIX "/usr"
- #define BINDIR "/usr/bin"
- #define LIBDIR "/usr/lib"
- #define LIBEXECDIR "/usr/libexec"
- #define SBINDIR "/usr/sbin"
- #define SYSCONFDIR "/usr/etc"
- #define INCLUDEDIR "/usr/include"
- #define DATAROOTDIR "/usr/share"
- #define DATADIR "/usr/share"
- #define DOCDIR "/usr/share/doc/airinv-0.1.2"
- #define MANDIR "/usr/share/man"
- #define INFODIR "/usr/share/info"
- #define HTMLDIR "/usr/share/doc/airinv-0.1.2/html"
- #define PDFDIR "/usr/share/doc/airinv-0.1.2/html"
- #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

25.149.1 Define Documentation

25.149.1.1 `#define PACKAGE "airinv"`

Definition at line 4 of file [airinv-paths.hpp](#).

25.149.1.2 `#define PACKAGE_NAME "AIRINV"`

Definition at line 5 of file [airinv-paths.hpp](#).

25.149.1.3 `#define PACKAGE_VERSION "0.1.2"`

Definition at line 6 of file [airinv-paths.hpp](#).

25.149.1.4 `#define PREFIXDIR "/usr"`

Definition at line 7 of file [airinv-paths.hpp](#).

25.149.1.5 `#define EXEC_PREFIX "/usr"`

Definition at line 8 of file [airinv-paths.hpp](#).

25.149.1.6 `#define BINDIR "/usr/bin"`

Definition at line 9 of file [airinv-paths.hpp](#).

25.149.1.7 `#define LIBDIR "/usr/lib"`

Definition at line 10 of file [airinv-paths.hpp](#).

25.149.1.8 `#define LIBEXECDIR "/usr/libexec"`

Definition at line 11 of file [airinv-paths.hpp](#).

25.149.1.9 `#define SBINDIR "/usr/sbin"`

Definition at line 12 of file [airinv-paths.hpp](#).

25.149.1.10 `#define SYSCONFDIR "/usr/etc"`

Definition at line 13 of file [airinv-paths.hpp](#).

25.149.1.11 `#define INCLUDEDIR "/usr/include"`

Definition at line 14 of file [airinv-paths.hpp](#).

25.149.1.12 `#define DATAROOTDIR "/usr/share"`

Definition at line 15 of file [airinv-paths.hpp](#).

25.149.1.13 `#define DATADIR "/usr/share"`

Definition at line 16 of file [airinv-paths.hpp](#).

25.149.1.14 `#define DOCDIR "/usr/share/doc/airinv-0.1.2"`

Definition at line 17 of file [airinv-paths.hpp](#).

25.149.1.15 `#define MANDIR "/usr/share/man"`

Definition at line 18 of file [airinv-paths.hpp](#).

25.149.1.16 `#define INFODIR "/usr/share/info"`

Definition at line 19 of file [airinv-paths.hpp](#).

25.149.1.17 `#define HTMLDIR "/usr/share/doc/airinv-0.1.2/html"`

Definition at line 20 of file [airinv-paths.hpp](#).

25.149.1.18 `#define PDFDIR "/usr/share/doc/airinv-0.1.2/html"`

Definition at line 21 of file [airinv-paths.hpp](#).

25.149.1.19 `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

Definition at line 22 of file [airinv-paths.hpp](#).

25.150 airinv-paths.hpp

```
00001 #ifndef __AIRINV_PATHS_HPP__
00002 #define __AIRINV_PATHS_HPP__
00003
00004 #define PACKAGE "airinv"
00005 #define PACKAGE_NAME "AIRINV"
00006 #define PACKAGE_VERSION "0.1.2"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/airinv-0.1.2"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/airinv-0.1.2/html"
00021 #define PDFDIR "/usr/share/doc/airinv-0.1.2/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __AIRINV_PATHS_HPP__
```

25.151 airinv/config/airinv-paths.hpp.in File Reference

Defines

- `#define __AIRINV_PATHS_HPP__`
- `#define PACKAGE "@PACKAGE@"`
- `#define PACKAGE_NAME "@PACKAGE_NAME@"`
- `#define PACKAGE_VERSION "@PACKAGE_VERSION@"`
- `#define PREFIXDIR "@prefix@"`
- `#define EXEC_PREFIX "@exec_prefix@"`
- `#define BINDIR "@bindir@"`
- `#define LIBDIR "@libdir@"`
- `#define LIBEXECDIR "@libexecdir@"`
- `#define SBINDIR "@sbindir@"`
- `#define SYSCONFDIR "@sysconfdir@"`
- `#define INCLUDEDIR "@includedir@"`
- `#define DATAROOTDIR "@datarootdir@"`
- `#define DATADIR "@datadir@"`
- `#define DOCDIR "@docdir@"`
- `#define MANDIR "@mandir@"`
- `#define INFODIR "@infodir@"`

- `#define HTMLDIR "@htmldir@"`
- `#define PDFDIR "@pdfdir@"`
- `#define STDAIR_SAMPLE_DIR "@sampledir@"`

25.151.1 Define Documentation

25.151.1.1 `#define __AIRINV_PATHS_HPP__`

Definition at line 2 of file [airinv-paths.hpp.in](#).

25.151.1.2 `#define PACKAGE "@PACKAGE@"`

Definition at line 4 of file [airinv-paths.hpp.in](#).

25.151.1.3 `#define PACKAGE_NAME "@PACKAGE_NAME@"`

Definition at line 5 of file [airinv-paths.hpp.in](#).

25.151.1.4 `#define PACKAGE_VERSION "@PACKAGE_VERSION@"`

Definition at line 6 of file [airinv-paths.hpp.in](#).

25.151.1.5 `#define PREFIXDIR "@prefix@"`

Definition at line 7 of file [airinv-paths.hpp.in](#).

25.151.1.6 `#define EXEC_PREFIX "@exec_prefix@"`

Definition at line 8 of file [airinv-paths.hpp.in](#).

25.151.1.7 `#define BINDIR "@bindir@"`

Definition at line 9 of file [airinv-paths.hpp.in](#).

25.151.1.8 `#define LIBDIR "@libdir@"`

Definition at line 10 of file [airinv-paths.hpp.in](#).

25.151.1.9 `#define LIBEXECDIR "@libexecdir@"`

Definition at line 11 of file [airinv-paths.hpp.in](#).

25.151.1.10 `#define SBINDIR "@sbindir@"`

Definition at line 12 of file [airinv-paths.hpp.in](#).

25.151.1.11 `#define SYSCONFDIR "@sysconfdir@"`

Definition at line 13 of file [airinv-paths.hpp.in](#).

25.151.1.12 `#define INCLUDEDIR "@includedir@"`

Definition at line 14 of file [airinv-paths.hpp.in](#).

25.151.1.13 `#define DATAROOTDIR "@datarootdir@"`

Definition at line 15 of file [airinv-paths.hpp.in](#).

25.151.1.14 `#define DATADIR "@datadir@"`

Definition at line 16 of file [airinv-paths.hpp.in](#).

25.151.1.15 `#define DOCDIR "@docdir@"`

Definition at line 17 of file [airinv-paths.hpp.in](#).

25.151.1.16 `#define MANDIR "@mandir@"`

Definition at line 18 of file [airinv-paths.hpp.in](#).

25.151.1.17 `#define INFODIR "@infodir@"`

Definition at line 19 of file [airinv-paths.hpp.in](#).

25.151.1.18 `#define HTMLDIR "@htmldir@"`

Definition at line 20 of file [airinv-paths.hpp.in](#).

25.151.1.19 `#define PDFDIR "@pdfdir@"`

Definition at line 21 of file [airinv-paths.hpp.in](#).

25.151.1.20 `#define STDAIR_SAMPLE_DIR "@sampledir@"`

Definition at line 22 of file [airinv-paths.hpp.in](#).

25.152 airinv-paths.hpp.in

```
00001 #ifndef __AIRINV_PATHS_HPP__
00002 #define __AIRINV_PATHS_HPP__
00003
00004 #define PACKAGE "@PACKAGE@"
00005 #define PACKAGE_NAME "@PACKAGE_NAME@"
00006 #define PACKAGE_VERSION "@PACKAGE_VERSION@"
00007 #define PREFIXDIR "@prefix@"
00008 #define EXEC_PREFIX "@exec_prefix@"
00009 #define BINDIR "@bindir@"
00010 #define LIBDIR "@libdir@"
00011 #define LIBEXECDIR "@libexecdir@"
00012 #define SBINDIR "@sbindir@"
00013 #define SYSCONFDIR "@sysconfdir@"
00014 #define INCLUDEDIR "@includedir@"
00015 #define DATAROOTDIR "@datarootdir@"
00016 #define DATADIR "@datadir@"
00017 #define DOCDIR "@docdir@"
00018 #define MANDIR "@mandir@"
00019 #define INFODIR "@infodir@"
00020 #define HTMLDIR "@htmldir@"
00021 #define PDFDIR "@pdfdir@"
00022 #define STDAIR_SAMPLE_DIR "@sampledir@"
00023
00024 #endif // __AIRINV_PATHS_HPP__
```

25.153 airinv/factory/FacAirinvMasterServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airinv/factory/FacAirinvMasterServiceContext.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.154 FacAirinvMasterServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirInv
00009 #include <airinv/factory/FacAirinvMasterServiceContext.hpp>
00010 #include <airinv/service/AIRINV_Master_ServiceContext.hpp>
00011
00012 namespace AIRINV {
00013
00014     FacAirinvMasterServiceContext* FacAirinvMasterServiceContext::_instance =
        NULL;
00015
00016 // //////////////////////////////////////
00017 FacAirinvMasterServiceContext::~FacAirinvMasterServiceContext() {
00018     _instance = NULL;
00019 }
00020
00021 // //////////////////////////////////////
00022 FacAirinvMasterServiceContext& FacAirinvMasterServiceContext::instance() {
00023
00024     if (_instance == NULL) {
00025         _instance = new FacAirinvMasterServiceContext();
00026         assert (_instance != NULL);
00027
00028         stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029     }
00030     return *_instance;
00031 }
00032
00033 // //////////////////////////////////////
00034 AIRINV_Master_ServiceContext& FacAirinvMasterServiceContext::create() {
00035     AIRINV_Master_ServiceContext* aAIRINV_Master_ServiceContext_ptr = NULL;
00036
00037     aAIRINV_Master_ServiceContext_ptr = new AIRINV_Master_ServiceContext();
00038     assert (aAIRINV_Master_ServiceContext_ptr != NULL);
00039
00040     // The new object is added to the Bom pool
00041     _pool.push_back (aAIRINV_Master_ServiceContext_ptr);
00042
00043     return *aAIRINV_Master_ServiceContext_ptr;
00044 }
00045
00046 }

```

25.155 airinv/factory/FacAirinvMasterServiceContext.hpp File Reference

```

#include <string>
#include <stdair/service/FacServiceAbstract.hpp>

```

Classes

- class [AIRINV::FacAirinvMasterServiceContext](#)
Factory for Bucket.

Namespaces

- namespace [AIRINV](#)

25.156 FacAirinvMasterServiceContext.hpp

```

00001 #ifndef __AIRINV_FAC_FACAIRINVMASERSERVICECONTEXT_HPP
00002 #define __AIRINV_FAC_FACAIRINVMASERSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/service/FacServiceAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015     class AIRINV_Master_ServiceContext;
00016
00020     class FacAirinvMasterServiceContext : public stdair::FacServiceAbstract {
00021     public:
00022
00026         static FacAirinvMasterServiceContext& instance();
00027
00032         ~FacAirinvMasterServiceContext();
00033
00037         AIRINV_Master_ServiceContext& create();
00038
00039     protected:
00044         FacAirinvMasterServiceContext() {}
00045
00046     private:
00048         static FacAirinvMasterServiceContext* _instance;
00049     };
00050
00051 }
00052 #endif // __AIRINV_FAC_FACAIRINVMASTERSERVICECONTEXT_HPP

```

25.157 airinv/factory/FacAirinvServiceContext.cpp File Reference

```

#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airinv/factory/FacAirinvServiceContext.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.158 FacAirinvServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirInv
00009 #include <airinv/factory/FacAirinvServiceContext.hpp>
00010 #include <airinv/service/AIRINV_ServiceContext.hpp>
00011
00012 namespace AIRINV {
00013
00014     FacAirinvServiceContext* FacAirinvServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacAirinvServiceContext::~FacAirinvServiceContext() {
00018         _instance = NULL;
00019     }
00020
00021     // //////////////////////////////////////
00022     FacAirinvServiceContext& FacAirinvServiceContext::instance() {
00023
00024         if (_instance == NULL) {
00025             _instance = new FacAirinvServiceContext();
00026             assert (_instance != NULL);
00027
00028             stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029         }
00030         return *_instance;
00031     }
00032

```

```

00033 // //////////////////////////////////////
00034 AIRINV_ServiceContext& FacAirinvServiceContext::create() {
00035     AIRINV_ServiceContext* aAIRINV_ServiceContext_ptr = NULL;
00036
00037     aAIRINV_ServiceContext_ptr = new AIRINV_ServiceContext();
00038     assert (aAIRINV_ServiceContext_ptr != NULL);
00039
00040     // The new object is added to the Bom pool
00041     _pool.push_back (aAIRINV_ServiceContext_ptr);
00042
00043     return *aAIRINV_ServiceContext_ptr;
00044 }
00045
00046 }

```

25.159 airinv/factory/FacAirinvServiceContext.hpp File Reference

```

#include <string>
#include <stdair/service/FacServiceAbstract.hpp>

```

Classes

- class [AIRINV::FacAirinvServiceContext](#)

Namespaces

- namespace [AIRINV](#)

25.160 FacAirinvServiceContext.hpp

```

00001 #ifndef __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP
00002 #define __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/service/FacServiceAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00014     class AIRINV_ServiceContext;
00015
00016     class FacAirinvServiceContext : public stdair::FacServiceAbstract {
00017     public:
00018
00019         static FacAirinvServiceContext& instance();
00020
00021         ~FacAirinvServiceContext();
00022
00023         AIRINV_ServiceContext& create();
00024
00025     protected:
00026         FacAirinvServiceContext() {}
00027
00028     private:
00029         static FacAirinvServiceContext* _instance;
00030     };
00031 }
00032
00033 #endif // __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP

```


25.161 airinv/factory/FacBomAbstract.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/functional/hash/hash.hpp>
#include <airinv/bom/BomAbstract.hpp>
#include <airinv/factory/FacBomAbstract.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.162 FacBomAbstract.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost (STL Extension)
00008 #include <boost/functional/hash/hash.hpp>
00009 // Airinv
00010 #include <airinv/bom/BomAbstract.hpp>
00011 #include <airinv/factory/FacBomAbstract.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 FacBomAbstract::~FacBomAbstract() {
00017     clean ();
00018 }
00019
00020 // //////////////////////////////////////
00021 void FacBomAbstract::clean() {
00022     for (BomPool_T::iterator itBom = _pool.begin();
00023          itBom != _pool.end(); itBom++) {
00024         BomAbstract* currentBom_ptr = *itBom;
00025         assert (currentBom_ptr != NULL);
00026
00027         delete (currentBom_ptr); currentBom_ptr = NULL;
00028     }
00029
00030     // Empty the pool of Factories
00031     _pool.clear();
00032 }
00033
00034 // //////////////////////////////////////
00035 std::size_t FacBomAbstract::getID (const BomAbstract* iBomAbstract_ptr) {
00036     const void* lPtr = iBomAbstract_ptr;
00037     boost::hash<const void*> ptr_hash;
00038     const std::size_t lID = ptr_hash (lPtr);
00039     return lID;
00040 }
00041
00042 // //////////////////////////////////////
00043 std::size_t FacBomAbstract::getID (const BomAbstract& iBomAbstract) {
00044     return getID (&iBomAbstract);
00045 }
00046
00047 // //////////////////////////////////////
00048 std::string FacBomAbstract::getIDString(const BomAbstract* iBomAbstract_ptr)
00049 {
00050     const std::size_t lID = getID (iBomAbstract_ptr);
00051     std::ostringstream oStr;
00052     oStr << lID;
00053     return oStr.str();
00054 }
00055
00056 // //////////////////////////////////////
00057 std::string FacBomAbstract::getIDString (const BomAbstract& iBomAbstract) {
00058     return getIDString (&iBomAbstract);
00059 }
00060 }
```

25.163 airinv/factory/FacBomAbstract.hpp File Reference

```
#include <string>
#include <vector>
```

Classes

- class [AIRINV::FacBomAbstract](#)

Namespaces

- namespace [AIRINV](#)

25.164 FacBomAbstract.hpp

```
00001 #ifndef __AIRINV_FAC_FACBOMABSTRACT_HPP
00002 #define __AIRINV_FAC_FACBOMABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010
00011 namespace AIRINV {
00012
00013     // Forward declarations
00014     class BomAbstract;
00015
00017     class FacBomAbstract {
00018     friend class FacSupervisor;
00019     public:
00020
00022         typedef std::vector<BomAbstract*> BomPool_T;
00023
00025         static std::size_t getID (const BomAbstract*);
00026
00028         static std::size_t getID (const BomAbstract&);
00029
00032         static std::string getIDString (const BomAbstract*);
00033
00036         static std::string getIDString (const BomAbstract&);
00037
00038     protected:
00041         FacBomAbstract() {}
00042         FacBomAbstract(const FacBomAbstract&) {}
00043
00045         virtual ~FacBomAbstract();
00046
00047     private:
00049         void clean();
00050
00051     protected:
00053         BomPool_T _pool;
00054     };
00055 }
00056 #endif // __AIRINV_FAC_FACBOMABSTRACT_HPP
```

25.165 airinv/factory/FacServiceAbstract.cpp File Reference

```
#include <cassert>
#include <airinv/service/ServiceAbstract.hpp>
#include <airinv/factory/FacServiceAbstract.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.166 FacServiceAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AIRINV
00007 #include <airinv/service/ServiceAbstract.hpp>
00008 #include <airinv/factory/FacServiceAbstract.hpp>
00009
00010 namespace AIRINV {
00011
00012 // //////////////////////////////////////
00013 FacServiceAbstract::~FacServiceAbstract() {
00014     clean ();
00015 }
00016
00017 // //////////////////////////////////////
00018 void FacServiceAbstract::clean() {
00019     for (ServicePool_T::iterator itService = _pool.begin();
00020          itService != _pool.end(); itService++) {
00021         ServiceAbstract* currentService_ptr = *itService;
00022         assert (currentService_ptr != NULL);
00023
00024         delete (currentService_ptr); currentService_ptr = NULL;
00025     }
00026
00027     // Empty the pool of Service Factories
00028     _pool.clear();
00029 }
00030
00031 }

```

25.167 airinv/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class [AIRINV::FacServiceAbstract](#)

Namespaces

- namespace [AIRINV](#)

25.168 FacServiceAbstract.hpp

```

00001 #ifndef __AIRINV_FAC_FACSERVICEABSTRACT_HPP
00002 #define __AIRINV_FAC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRINV {
00011
00012 // Forward declarations
00013 class ServiceAbstract;
00014
00016 class FacServiceAbstract {
00017 public:
00018
00020     typedef std::vector<ServiceAbstract*> ServicePool_T;

```

```

00021
00023     virtual ~FacServiceAbstract();
00024
00026     void clean();
00027
00028     protected:
00031         FacServiceAbstract() {}
00032
00034         ServicePool_T _pool;
00035     };
00036
00037 }
00038 #endif // __AIRINV_FAC_FACSERVICEABSTRACT_HPP

```

25.169 airinv/factory/FacSupervisor.cpp File Reference

```

#include <cassert>
#include <airinv/factory/FacBomAbstract.hpp>
#include <airinv/factory/FacServiceAbstract.hpp>
#include <airinv/factory/FacSupervisor.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.170 FacSupervisor.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AIRINV
00007 #include <airinv/factory/FacBomAbstract.hpp>
00008 #include <airinv/factory/FacServiceAbstract.hpp>
00009 #include <airinv/factory/FacSupervisor.hpp>
00010
00011 namespace AIRINV {
00012
00013     FacSupervisor* FacSupervisor::_instance = NULL;
00014
00015     // //////////////////////////////////////
00016     FacSupervisor::FacSupervisor() {
00017     }
00018
00019     // //////////////////////////////////////
00020     FacSupervisor& FacSupervisor::instance() {
00021         if (_instance == NULL) {
00022             _instance = new FacSupervisor();
00023         }
00024
00025         return *_instance;
00026     }
00027
00028     // //////////////////////////////////////
00029     void FacSupervisor::
00030     registerBomFactory (FacBomAbstract* ioFacBomAbstract_ptr) {
00031         _bomPool.push_back (ioFacBomAbstract_ptr);
00032     }
00033
00034     // //////////////////////////////////////
00035     void FacSupervisor::
00036     registerServiceFactory (FacServiceAbstract* ioFacServiceAbstract_ptr) {
00037         _svcPool.push_back (ioFacServiceAbstract_ptr);
00038     }
00039
00040     // //////////////////////////////////////
00041     FacSupervisor::~~FacSupervisor() {
00042         cleanBomLayer();
00043         cleanServiceLayer();
00044     }
00045
00046     // //////////////////////////////////////
00047     void FacSupervisor::cleanBomLayer() {
00048         for (BomFactoryPool_T::const_iterator itFactory = _bomPool.begin();

```

```

00049         itFactory != _bomPool.end(); itFactory++) {
00050             const FacBomAbstract* currentFactory_ptr = *itFactory;
00051             assert (currentFactory_ptr != NULL);
00052
00053             delete (currentFactory_ptr); currentFactory_ptr = NULL;
00054         }
00055
00056         // Empty the pool of Bom Factories
00057         _bomPool.clear();
00058     }
00059
00060     // //////////////////////////////////////
00061     void FacSupervisor::cleanServiceLayer() {
00062         for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
00063             itFactory != _svcPool.end(); itFactory++) {
00064             const FacServiceAbstract* currentFactory_ptr = *itFactory;
00065             assert (currentFactory_ptr != NULL);
00066
00067             delete (currentFactory_ptr); currentFactory_ptr = NULL;
00068         }
00069
00070         // Empty the pool of Service Factories
00071         _svcPool.clear();
00072     }
00073
00074     // //////////////////////////////////////
00075     void FacSupervisor::cleanFactory () {
00076         if (_instance != NULL) {
00077             _instance->cleanBomLayer();
00078             _instance->cleanServiceLayer();
00079         }
00080         delete (_instance); _instance = NULL;
00081     }
00082
00083 }

```

25.171 airinv/factory/FacSupervisor.hpp File Reference

```
#include <vector>
```

Classes

- class [AIRINV::FacSupervisor](#)

Namespaces

- namespace [AIRINV](#)

25.172 FacSupervisor.hpp

```

00001 #ifndef __AIRINV_FAC_FACSUPERVISOR_HPP
00002 #define __AIRINV_FAC_FACSUPERVISOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRINV {
00011
00012     // Forward declarations
00013     class FacBomAbstract;
00014     class FacServiceAbstract;
00015
00017     class FacSupervisor {
00018     public:
00019
00021         typedef std::vector<FacBomAbstract*> BomFactoryPool_T;
00022         typedef std::vector<FacServiceAbstract*> ServiceFactoryPool_T;
00023
00027         static FacSupervisor& instance();
00028

```

```

00033     void registerBomFactory (FacBomAbstract*);
00034
00039     void registerServiceFactory (FacServiceAbstract*);
00040
00044     void cleanBomLayer();
00045
00049     void cleanServiceLayer();
00050
00053     static void cleanFactory ();
00054
00058     ~FacSupervisor();
00059
00060
00061 protected:
00065     FacSupervisor ();
00066     FacSupervisor (const FacSupervisor&) {}
00067
00068
00069 private:
00071     static FacSupervisor* _instance;
00072
00074     BomFactoryPool_T _bomPool;
00075
00077     ServiceFactoryPool_T _svcPool;
00078 };
00079 }
00080 #endif // __AIRINV_FAC_FACSUPERVISOR_HPP

```

25.173 airinv/FlightRequestStatus.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [AIRINV::FlightRequestStatus](#)

Namespaces

- namespace [AIRINV](#)

25.174 FlightRequestStatus.hpp

```

00001 #ifndef __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP
00002 #define __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015     struct FlightRequestStatus : public stdair::StructAbstract {
00016     public:
00017         typedef enum {
00018             OK = 0,
00019             NOT_FOUND,
00020             INTERNAL_ERROR,
00021             LAST_VALUE
00022         } EN_FlightRequestStatus;
00023
00025         static const std::string& getLabel (const EN_FlightRequestStatus&);
00026
00028         static const std::string& getCodeLabel (const EN_FlightRequestStatus&);
00029
00031         static std::string describeLabels();
00032
00034         EN_FlightRequestStatus getCode() const;
00035

```

```

00037     const std::string describe() const;
00038
00039
00040 public:
00042     FlightRequestStatus (const EN_FlightRequestStatus&);
00044     FlightRequestStatus (const std::string& iCode);
00045
00046
00047 private:
00049     static const std::string _labels[LAST_VALUE];
00051     static const std::string _codeLabels[LAST_VALUE];
00052
00053
00054 private:
00055     // ////////// Attributes //////////
00057     EN_FlightRequestStatus _code;
00058 };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP

```

25.175 airinv/server/AirInvClient.cpp File Reference

```

#include <string>
#include <iostream>
#include <zmq.hpp>

```

Functions

- int [main](#) (int argc, char *argv[])

25.175.1 Function Documentation

25.175.1.1 int main (int argc, char * argv[])

Definition at line 11 of file [AirInvClient.cpp](#).

25.176 AirInvClient.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <string>
00006 #include <iostream>
00007 // ZeroMQ
00008 #include <zmq.hpp>
00009
00010 // ////////////////////////////////////// M A I N //////////////////////////////////////
00011 int main (int argc, char* argv[]) {
00012     // Prepare our context and socket
00013     zmq::context_t context (1);
00014     zmq::socket_t socket (context, ZMQ_REQ);
00015
00016     std::cout << "Connecting to hello world server..." << std::endl;
00017     socket.connect ("tcp://localhost:5555");
00018
00019     // Do 10 requests, waiting each time for a response
00020     for (int request_nbr = 0; request_nbr != 10; request_nbr++) {
00021         zmq::message_t request (6);
00022         memcpy ((void *) request.data (), "Hello", 5);
00023         std::cout << "Sending Hello " << request_nbr << "..." << std::endl;
00024         socket.send (request);
00025
00026         // Get the reply.
00027         zmq::message_t reply;
00028         socket.recv (&reply);
00029         std::cout << "Received World " << request_nbr << std::endl;
00030     }
00031     return 0;
00032 }

```

25.177 airinv/server/AirInvClient_ASIO.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/array.hpp>
```

Functions

- `int main (int argc, char *argv[])`

25.177.1 Function Documentation

25.177.1.1 `int main (int argc, char * argv[])`

Definition at line 14 of file [AirInvClient_ASIO.cpp](#).

25.178 AirInvClient_ASIO.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <iostream>
00007 #include <string>
00008 // Boost.ASIO
00009 #include <boost/asio.hpp>
00010 // Boost.Array
00011 #include <boost/array.hpp>
00012
00013 // /////////// M A I N ///////////
00014 int main (int argc, char* argv[]) {
00015
00016     // Host name
00017     std::string lHostname = "localhost";
00018
00019     // Service name (as specified within /etc/services)
00020     // The "aria" service corresponds to the port 2624
00021     const std::string lServiceName = "aria";
00022
00023     try {
00024
00025         if (argc >= 2) {
00026             lHostname = argv[1];
00027         }
00028
00029         boost::asio::io_service lIOService;
00030
00031         boost::asio::ip::tcp::resolver lResolver (lIOService);
00032
00033         boost::asio::ip::tcp::resolver::query lQuery (lHostname, lServiceName);
00034
00035         boost::asio::ip::tcp::resolver::iterator itEndPoint =
00036             lResolver.resolve (lQuery);
00037         boost::asio::ip::tcp::resolver::iterator lEnd;
00038
00039         boost::asio::ip::tcp::socket lSocket (lIOService);
00040         boost::system::error_code lError = boost::asio::error::host_not_found;
00041
00042         //
00043         while (lError && itEndPoint != lEnd) {
00044             const boost::asio::ip::tcp::endpoint lEndPoint = *itEndPoint;
00045
00046             // DEBUG
00047             std::cout << "Testing end point: " << std::endl;
00048
00049             lSocket.close();
00050             lSocket.connect (lEndPoint, lError);
00051             ++itEndPoint;
00052         }
00053     }
```



```

00054     //
00055     if (lError) {
00056         throw boost::system::system_error (lError);
00057     }
00058     assert (!lError);
00059
00060     // DEBUG
00061     const boost::asio::ip::tcp::endpoint lValidEndPoint;
00062     std::cout << "Valid end point: " << lValidEndPoint << std::endl;
00063
00064     // Send a message to the server
00065     const std::string lMessage ("Hello AirInv Server!");
00066     boost::asio::write (lSocket, boost::asio::buffer (lMessage),
00067         boost::asio::transfer_all(), lError);
00068
00069     // Read the reply from the server
00070     boost::array<char, 256> lBuffer;
00071
00072     size_t lLength = lSocket.read_some (boost::asio::buffer(lBuffer), lError);
00073
00074     // Some other error than connection closed cleanly by peer
00075     if (lError && lError != boost::asio::error::eof) {
00076         throw boost::system::system_error (lError);
00077     }
00078
00079     // DEBUG
00080     std::cout << "Reply from the server: ";
00081     std::cout.write (lBuffer.data(), lLength);
00082     std::cout << std::endl;
00083
00084 } catch (std::exception& lException) {
00085     std::cerr << lException.what() << std::endl;
00086 }
00087
00088 return 0;
00089 }

```

25.179 airinv/server/AirInvServer.cpp File Reference

25.180 AirInvServer.cpp

```

00001
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 #include <fstream>
00012 #include <string>
00013 #include <unistd.h>
00014 // Boost (Extended STL)
00015 #include <boost/program_options.hpp>
00016 #include <boost/tokenizer.hpp>
00017 // ZeroMQ
00018 #include <zmq.hpp>
00019 // StdAir
00020 #include <stdair/basic/BasLogParams.hpp>
00021 #include <stdair/basic/BasDBParams.hpp>
00022 #include <stdair/bom/BomJSONImport.hpp>
00023 #include <stdair/bom/BomJSONExport.hpp>
00024 #include <stdair/service/Logger.hpp>
00025 // AirInvServer
00026 #include <airinv/config/airinv-paths.hpp>
00027 #include <airinv/AIRINV_Master_Service.hpp>
00028
00029 // ////////// Type definitions //////////
00030 typedef unsigned int ServerPort_T;
00031
00032 // ////////// Constants //////////
00033 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinvServer.log");
00034
00037 const std::string K_AIRINV_DEFAULT_SERVER_PROTOCOL ("tcp://");
00038
00040 const std::string K_AIRINV_DEFAULT_SERVER_ADDRESS ("*");
00041
00043 const ServerPort_T K_AIRINV_DEFAULT_SERVER_PORT (5555);
00044
00046 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00047     "/invdump01.csv");
00049 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00050     "/schedule01.csv");
00052 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR

```

```

00053                                     "/ond01.csv");
00054
00056 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00057                                     "/yield01.csv");
00058
00063 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00064
00069 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00070
00074 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00075
00079 struct Command_T {
00080     typedef enum {
00081         NOP = 0,
00082         QUIT,
00083         DISPLAY,
00084         SELL,
00085         LAST_VALUE
00086     } Type_T;
00087 };
00088
00089 // ////////// Parsing of Options & Configuration //////////
00090 // A helper function to simplify the main part.
00091 template<class T> std::ostream& operator<< (std::ostream& os,
00092     const std::vector<T>& v) {
00093     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00094     return os;
00095 }
00096
00098 int readConfiguration (int argc, char* argv[], std::string& ioServerProtocol,
00099     std::string& ioServerAddress, ServerPort_T& ioServerPort
00100 ,
00101     bool& ioIsBuiltin, bool& ioIsForSchedule,
00102     stdair::Filename_T& ioInventoryFilename,
00103     stdair::Filename_T& ioScheduleInputFilename,
00104     stdair::Filename_T& ioODInputFilename,
00105     stdair::Filename_T& ioYieldInputFilename,
00106     std::string& ioLogFilename) {
00107     // Default for the built-in input
00108     ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00109     // Default for the inventory or schedule option
00110     ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00111
00112     // Declare a group of options that will be allowed only on command line
00113     boost::program_options::options_description generic ("Generic options");
00114     generic.add_options()
00115         ("prefix", "print installation prefix")
00116         ("version,v", "print version string")
00117         ("help,h", "produce help message");
00118
00119     // Declare a group of options that will be allowed both on command
00120     // line and in config file
00121
00122     boost::program_options::options_description config ("Configuration");
00123     config.add_options()
00124         ("builtin,b",
00125         "The sample BOM tree can be either built-in or parsed from an input file.
00126         That latter must then be given with the -i/--inventory or -s/--schedule option")
00127         ("for_schedule,f",
00128         "The BOM tree should be built from a schedule file (instead of from an
00129         inventory dump)")
00130         ("inventory,i",
00131         boost::program_options::value< std::string >(&ioInventoryFilename)->
00132         default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00133         "(CVS) input file for the inventory")
00134         ("schedule,s",
00135         boost::program_options::value< std::string >(&ioScheduleInputFilename)->
00136         default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00137         "(CVS) input file for the schedule")
00138         ("ond,o",
00139         boost::program_options::value< std::string >(&ioODInputFilename)->
00140         default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00141         "(CVS) input file for the O&D")
00142         ("yield,y",
00143         boost::program_options::value< std::string >(&ioYieldInputFilename)->
00144         default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00145         "(CVS) input file for the yield")
00146         ("protocol,t",
00147         boost::program_options::value< std::string >(&ioServerProtocol)->
00148         default_value(K_AIRINV_DEFAULT_SERVER_PROTOCOL),
00149         "Server protocol")
00150         ("address,a",
00151         boost::program_options::value< std::string >(&ioServerAddress)->
00152         default_value(K_AIRINV_DEFAULT_SERVER_ADDRESS),
00153         "Server address")
00154         ("port,p",

```

```

00147     boost::program_options::value< ServerPort_T >(&ioServerPort)->
default_value(K_AIRINV_DEFAULT_SERVER_PORT),
00148     "Server port")
00149     ("log,l",
00150     boost::program_options::value< std::string >(&ioLogFilename)->
default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00151     "Filename for the output logs")
00152     ;
00153
00154     // Hidden options, will be allowed both on command line and
00155     // in config file, but will not be shown to the user.
00156     boost::program_options::options_description hidden ("Hidden options");
00157     hidden.add_options()
00158         ("copyright",
00159         boost::program_options::value< std::vector<std::string> >(),
00160         "Show the copyright (license)");
00161
00162     boost::program_options::options_description cmdline_options;
00163     cmdline_options.add(generic).add(config).add(hidden);
00164
00165     boost::program_options::options_description config_file_options;
00166     config_file_options.add(config).add(hidden);
00167     boost::program_options::options_description visible ("Allowed options");
00168     visible.add(generic).add(config);
00169
00170     boost::program_options::positional_options_description p;
00171     p.add ("copyright", -1);
00172
00173     boost::program_options::variables_map vm;
00174     boost::program_options::
00175         store (boost::program_options::command_line_parser (argc, argv).
00176         options (cmdline_options).positional(p).run(), vm);
00177
00178     std::ifstream ifs ("airinvServer.cfg");
00179     boost::program_options::store (parse_config_file (ifs, config_file_options),
00180     vm);
00181     boost::program_options::notify (vm);
00182
00183     if (vm.count ("help")) {
00184         std::cout << visible << std::endl;
00185         return K_AIRINV_EARLY_RETURN_STATUS;
00186     }
00187
00188     if (vm.count ("version")) {
00189         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00190         return K_AIRINV_EARLY_RETURN_STATUS;
00191     }
00192
00193     if (vm.count ("prefix")) {
00194         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00195         return K_AIRINV_EARLY_RETURN_STATUS;
00196     }
00197
00198     if (vm.count ("protocol")) {
00199         ioServerProtocol = vm["protocol"].as< std::string >();
00200         std::cout << "Server protocol is: " << ioServerProtocol << std::endl;
00201     }
00202
00203     if (vm.count ("address")) {
00204         ioServerAddress = vm["address"].as< std::string >();
00205         std::cout << "Server address is: " << ioServerAddress << std::endl;
00206     }
00207
00208     if (vm.count ("port")) {
00209         ioServerPort = vm["port"].as< ServerPort_T >();
00210         std::cout << "Server port is: " << ioServerPort << std::endl;
00211     }
00212
00213     if (vm.count ("builtin")) {
00214         ioIsBuiltin = true;
00215     }
00216     const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00217     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00218
00219     if (vm.count ("for_schedule")) {
00220         ioIsForSchedule = true;
00221     }
00222     const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00223     std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00224         << std::endl;
00225
00226     if (ioIsBuiltin == false) {
00227
00228         if (ioIsForSchedule == false) {
00229             // The BOM tree should be built from parsing an inventory dump
00230             if (vm.count ("inventory")) {
00231                 ioInventoryFilename = vm["inventory"].as< std::string >();

```

```

00232         std::cout << "Input inventory filename is: " << ioInventoryFilename
00233                 << std::endl;
00234
00235     } else {
00236         // The built-in option is not selected. However, no inventory dump
00237         // file is specified
00238         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00239                 << " -f/--for_schedule and -s/--schedule options "
00240                 << "must be specified" << std::endl;
00241     }
00242
00243 } else {
00244     // The BOM tree should be built from parsing a schedule (and O&D) file
00245     if (vm.count ("schedule")) {
00246         ioScheduleInputFilename = vm["schedule"].as< std::string >();
00247         std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00248                 << std::endl;
00249
00250     } else {
00251         // The built-in option is not selected. However, no schedule file
00252         // is specified
00253         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00254                 << " -f/--for_schedule and -s/--schedule options "
00255                 << "must be specified" << std::endl;
00256     }
00257
00258     if (vm.count ("ond")) {
00259         ioODInputFilename = vm["ond"].as< std::string >();
00260         std::cout << "Input O&D filename is: " << ioODInputFilename <<
std::endl;
00261     }
00262
00263     if (vm.count ("yield")) {
00264         ioYieldInputFilename = vm["yield"].as< std::string >();
00265         std::cout << "Input yield filename is: " << ioYieldInputFilename <<
std::endl;
00266     }
00267 }
00268 }
00269
00270 if (vm.count ("log")) {
00271     ioLogFilename = vm["log"].as< std::string >();
00272     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00273 }
00274
00275 return 0;
00276 }
00277
00278
00279 // //////////// Utility functions on top of the ZeroMQ library ////////////
00283 static std::string s_recv (zmq::socket_t& socket) {
00284     zmq::message_t message;
00285     socket.recv (&message);
00286
00287     return std::string (static_cast<char*> (message.data()), message.size());
00288 }
00289
00293 static bool s_send (zmq::socket_t& socket, const std::string& string) {
00294     zmq::message_t message (string.size());
00295     memcpy (message.data(), string.data(), string.size());
00296
00297     bool rc = socket.send (message);
00298     return rc;
00299 }
00300
00301
00302 // ////////////////////////////////// M A I N //////////////////////////////////
00303 int main (int argc, char* argv[]) {
00304
00305     // Server parameters (for ZeroMQ)
00306     std::string ioServerProtocol;
00307     std::string ioServerAddress;
00308     ServerPort_T ioServerPort;
00309
00310     // State whether the BOM tree should be built-in or parsed from an
00311     // input file
00312     bool isBuiltin;
00313     bool isForSchedule;
00314
00315     // Input file names
00316     stdair::Filename_T lInventoryFilename;
00317     stdair::Filename_T lScheduleInputFilename;
00318     stdair::Filename_T lODInputFilename;
00319     stdair::Filename_T lYieldInputFilename;
00320
00321     // Output log File
00322     stdair::Filename_T lLogFilename;

```

```

00323
00324 // Call the command-line option parser
00325 const int lOptionParserStatus =
00326     readConfiguration (argc, argv, ioServerProtocol, ioServerAddress,
00327                         ioServerPort, isBuiltin, isForSchedule,
00328                         lInventoryFilename, lScheduleInputFilename,
00329                         lODInputFilename, lYieldInputFilename, lLogFilename);
00330
00331 if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00332     return 0;
00333 }
00334
00335 // Set the log parameters
00336 std::ofstream logOutputFile;
00337 // Open and clean the log outputfile
00338 logOutputFile.open (lLogFilename.c_str());
00339 logOutputFile.clear();
00340
00341 // Initialise the inventory service
00342 const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00343 AIRINV::AIRINV_Master_Service airinvService (lLogParams);
00344
00345 // DEBUG
00346 STDAIR_LOG_DEBUG ("Initialisation of the AirInv server");
00347
00348 // Check whether or not a (CSV) input file should be read
00349 if (isBuiltin == true) {
00350
00351     // Build the sample BOM tree for RMOL
00352     airinvService.buildSampleBom();
00353
00354 } else {
00355     if (isForSchedule == true) {
00356         // Build the BOM tree from parsing a schedule file (and O&D list)
00357         AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00358         airinvService.parseAndLoad (lScheduleInputFilename, lODInputFilename,
00359                                     lYieldFilePath);
00360     } else {
00361         // Build the BOM tree from parsing an inventory dump file
00362         airinvService.parseAndLoad (lInventoryFilename);
00363     }
00364 }
00365
00366 // Build the connection string (e.g., "tcp://*:5555", which is the default)
00367 std::ostringstream oZeroMQBindStream;
00368 oZeroMQBindStream << ioServerProtocol << ioServerAddress
00369                     << ":" << ioServerPort;
00370 const std::string lZeroMQBindString (oZeroMQBindStream.str());
00371
00372 // Prepare the context and socket of the server
00373 zmq::context_t context (1);
00374 zmq::socket_t socket (context, ZMQ_REP);
00375 socket.bind (lZeroMQBindString.c_str());
00376
00377 // DEBUG
00378 STDAIR_LOG_DEBUG ("The AirInv server is ready to receive requests...");
00379
00380 while (true) {
00381
00382     // Wait for next request from client, which is expected to give
00383     // the JSON-ified details of the requested flight-date
00384     const std::string& lFlightDateKeyJSONString = s_recv (socket);
00385
00386     // DEBUG
00387     STDAIR_LOG_DEBUG ("Received: '" << lFlightDateKeyJSONString << "'");
00388
00389     // Extract, from the JSON-ified string an airline code
00390     stdair::AirlineCode_T lAirlineCode;
00391     stdair::BomJSONImport::jsonImportInventoryKey (lFlightDateKeyJSONString,
00392                                                    lAirlineCode);
00393
00394     // Extract, from the JSON-ified string a flight number and a departure date
00395     stdair::FlightNumber_T lFlightNumber;
00396     stdair::Date_T lDate;
00397     stdair::BomJSONImport::jsonImportFlightDateKey (lFlightDateKeyJSONString,
00398                                                     lFlightNumber, lDate);
00399
00400     // DEBUG
00401     STDAIR_LOG_DEBUG ("=> airline code = '" << lAirlineCode
00402                       << "', flight number = '" << lFlightNumber
00403                       << "', departure date = '" << lDate << "'");
00404
00405     // DEBUG: Display the flight-date dump
00406     const std::string& lFlightDateCSVDump =
00407         airinvService.csvDisplay (lAirlineCode, lFlightNumber, lDate);
00408     STDAIR_LOG_DEBUG (std::endl << lFlightDateCSVDump);
00409

```

```

00410
00411 // Dump the full details of the flight-date into the JSON-ified flight-date
00412 const std::string& lFlightDateJSONDump =
00413     airinvService.jsonExport (lAirlineCode, lFlightNumber, lDate);
00414
00415 // DEBUG
00416 STDAIR_LOG_DEBUG ("Send: '" << lFlightDateJSONDump << "'");
00417
00418 // Send back the flight-date details to the client
00419 s_send (socket, lFlightDateJSONDump);
00420 }
00421
00422 return 0;
00423 }
00424

```

25.181 airinv/server/AirInvServer.hpp File Reference

```

#include <string>
#include <vector>
#include <boost/asio.hpp>
#include <boost/noncopyable.hpp>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <airinv/server/Connection.hpp>
#include <airinv/server/RequestHandler.hpp>

```

Classes

- class [AIRINV::AirInvServer](#)

Namespaces

- namespace [AIRINV](#)

25.182 AirInvServer.hpp

```

00001 #ifndef __AIRINV_SVR_AIRINVSERVICES_HPP
00002 #define __AIRINV_SVR_AIRINVSERVICES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // Boost
00011 #include <boost/asio.hpp>
00012 #include <boost/noncopyable.hpp>
00013 #include <boost/shared_ptr.hpp>
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 // AirInv
00017 #include <airinv/server/Connection.hpp>
00018 #include <airinv/server/RequestHandler.hpp>
00019
00020 namespace AIRINV {
00021
00022 class AirInvServer : private boost::noncopyable {
00023 public:
00024 // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00025 AirInvServer (const std::string& address, const std::string& port,
00026               const stdair::AirlineCode_T& iAirlineCode,
00027               std::size_t thread_pool_size);
00028 ~AirInvServer ();
00029
00030 public:
00031 // ////////////////////////////////// Business Methods //////////////////////////////////
00032 void run();

```

```

00040
00042     void stop();
00043
00044
00045 private:
00046     // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00048     AirInvServer();
00049     AirInvServer(const AirInvServer&);
00050
00051
00052 private:
00053     // ////////////////////////////////// Attributes //////////////////////////////////
00055     void handleAccept (const boost::system::error_code& e);
00056
00058     std::size_t _threadPoolSize;
00059
00061     boost::asio::io_service _ioService;
00062
00064     boost::asio::ip::tcp::acceptor _acceptor;
00065
00067     ConnectionShrPtr_T _newConnection;
00068
00070     RequestHandler _requestHandler;
00071 };
00072
00073 }
00074 #endif // __AIRINV_SVR_AIRINVSERVER_HPP

```

25.183 airinv/server/AirInvServer_ASIO.cpp File Reference

```

#include <cassert>
#include <boost/thread.hpp>
#include <boost/bind.hpp>
#include <airinv/server/AirInvServer.hpp>

```

Namespaces

- namespace [AIRINV](#)

Typedefs

- typedef boost::shared_ptr
< boost::thread > [AIRINV::ThreadShrPtr_T](#)
- typedef std::vector
< ThreadShrPtr_T > [AIRINV::ThreadShrPtrList_T](#)

25.184 AirInvServer_ASIO.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/thread.hpp>
00008 #include <boost/bind.hpp>
00009 // AirInv
00010 #include <airinv/server/AirInvServer.hpp>
00011
00012 namespace AIRINV {
00013
00014     // Type definitions
00015     typedef boost::shared_ptr<boost::thread> ThreadShrPtr_T;
00016     typedef std::vector<ThreadShrPtr_T> ThreadShrPtrList_T;
00017
00018
00019     // //////////////////////////////////////
00020     AirInvServer::AirInvServer (const std::string& address,
00021                                 const std::string& port,
00022                                 const stdair::AirlineCode_T& iAirlineCode,

```

```

00023         std::size_t iThreadPoolSize)
00024     : _threadPoolSize (iThreadPoolSize), _acceptor (_ioService),
00025       _newConnection (new Connection (_ioService, _requestHandler)),
00026       _requestHandler (iAirlineCode) {
00027
00028         // Open the acceptor with the option to reuse the address
00029         // (i.e. SO_REUSEADDR).
00030         boost::asio::ip::tcp::resolver resolver (_ioService);
00031         boost::asio::ip::tcp::resolver::query query (address, port);
00032         boost::asio::ip::tcp::endpoint endpoint = *resolver.resolve(query);
00033
00034         _acceptor.open (endpoint.protocol());
00035         _acceptor.set_option (boost::asio::ip::tcp::acceptor::reuse_address(true));
00036         _acceptor.bind (endpoint);
00037         _acceptor.listen();
00038
00039         assert (_newConnection != NULL);
00040         _acceptor.async_accept (_newConnection->socket(),
00041                                boost::bind (&AirInvServer::handleAccept, this,
00042                                                boost::asio::placeholders::error));
00043     }
00044
00045     // //////////////////////////////////////
00046     AirInvServer::~AirInvServer () {
00047     }
00048
00049     // //////////////////////////////////////
00050     void AirInvServer::run() {
00051         // Create a pool of threads to run all of the io_services.
00052         ThreadShrPtrList_T lThreadList;
00053
00054         for (std::size_t itThread = 0; itThread != _threadPoolSize; ++itThread) {
00055             ThreadShrPtr_T lThread (new boost::thread (boost::bind (&
00056                 boost::asio::io_service::run,
00057                                     &_ioService)));
00058             lThreadList.push_back (lThread);
00059         }
00060
00061         // Wait for all threads in the pool to exit.
00062         for (std::size_t itThread = 0; itThread != lThreadList.size(); ++itThread)
00063         {
00064             boost::shared_ptr<boost::thread> lThread_ptr = lThreadList.at (itThread);
00065             assert (lThread_ptr != NULL);
00066             lThread_ptr->join();
00067         }
00068
00069         // //////////////////////////////////////
00070         void AirInvServer::stop() {
00071             _ioService.stop();
00072         }
00073
00074         // //////////////////////////////////////
00075         void AirInvServer::handleAccept (const boost::system::error_code& iError) {
00076             if (!iError) {
00077                 assert (_newConnection != NULL);
00078
00079                 // The Connection object now takes in charge reading an incoming
00080                 // message from the socket, and writing back a message.
00081                 _newConnection->start();
00082
00083                 // The (Boost) shared pointer is resetted to a newly allocated Connection
00084                 // object. As the older Connection object is no longer pointed to, it is
00085                 // deleted by the shared pointer mechanism.
00086                 _newConnection.reset (new Connection (_ioService, _requestHandler));
00087
00088                 _acceptor.async_accept (_newConnection->socket(),
00089                                         boost::bind (&AirInvServer::handleAccept, this,
00090                                                         boost::asio::placeholders::error));
00091             }
00092         }
00093     }
00094
00095 }

```

25.185 airinv/server/BomPropertyTree.cpp File Reference

```

#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/foreach.hpp>
#include <airinv/server/BomPropertyTree.hpp>

```


Namespaces

- namespace `stdair`

Forward declarations.

25.186 BomPropertyTree.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // Boost Property Tree
00005 #include <boost/property_tree/ptree.hpp>
00006 #include <boost/property_tree/json_parser.hpp>
00007 // Boost ForEach
00008 #include <boost/foreach.hpp>
00009 // AirInvServer
00010 #include <airinv/server/BomPropertyTree.hpp>
00011
00012 namespace bpt = boost::property_tree;
00013
00014 namespace stdair {
00015
00016     // Loads BomPropertyTree structure from the specified JSON file
00017     void BomPropertyTree::load (const std::string& iBomTree) {
00018         // Create an empty property tree object
00019         bpt::ptree pt;
00020
00021         // Load the JSON formatted string into the property tree. If reading fails
00022         // (cannot open stream, parse error), an exception is thrown.
00023         std::istringstream iStr (iBomTree);
00024         read_json (iStr, pt);
00025
00026         // Get the airline_code and store it in the _airlineCode variable.
00027         // Note that we construct the path to the value by separating
00028         // the individual keys with dots. If dots appear in the keys,
00029         // a path type with a different separator can be used.
00030         // If the flight_date.airline_code key is not found, an exception is
00031         // thrown.
00032         _airlineCode = pt.get<stdair::AirlineCode_T> ("flight_date.airline_code");
00033
00034         // Get the departure_date and store it in the _departureDate variable.
00035         // This is another version of the get method: if the value is
00036         // not found, the default value (specified by the second
00037         // parameter) is returned instead. The type of the value
00038         // extracted is determined by the type of the second parameter,
00039         // so we can simply write get(...) instead of get<int>(...).
00040         _flightNumber =
00041             pt.get<stdair::FlightNumber_T> ("flight_date.flight_number", 100);
00042
00043         const std::string& lDepartureDateStr =
00044             pt.get<std::string> ("flight_date.departure_date");
00045         _departureDate = boost::gregorian::from_simple_string (lDepartureDateStr);
00046
00047         // Iterate over the flight_date.airport_codes section and store all found
00048         // codes in the _airportCodeList set. The get_child() function
00049         // returns a reference to the child at the specified path; if
00050         // there is no such child, it throws. Property tree iterators
00051         // are models of BidirectionalIterator.
00052         /*
00053         BOOST_FOREACH (bpt::ptree::value_type &v,
00054             pt.get_child ("flight_date.airport_codes")) {
00055             _airportCodeList.insert (v.second.data());
00056         }
00057         */
00058
00059         // Saves the BomPropertyTree structure to the specified JSON file
00060         std::string BomPropertyTree::save() const {
00061             std::ostringstream oStr;
00062
00063             // Create an empty property tree object
00064             bpt::ptree pt;
00065
00066             // Put airline code in property tree
00067             pt.put ("flight_date.airline_code", _airlineCode);
00068
00069             // Put flight number level in property tree
00070             pt.put ("flight_date.flight_number", _flightNumber);

```

```

00071
00072 // Put the flight departure date in property tree
00073 const std::string& lDepartureDateStr =
00074     boost::gregorian::to_simple_string (_departureDate);
00075 pt.put ("flight_date.departure_date", lDepartureDateStr);
00076
00077 // Iterate over the airport codes in the set and put them in the
00078 // property tree. Note that the put function places the new
00079 // key at the end of the list of keys. This is fine most of
00080 // the time. If you want to place an item at some other place
00081 // (i.e. at the front or somewhere in the middle), this can
00082 // be achieved using a combination of the insert and put_own
00083 // functions.
00084 bpt::ptree lAirportCodeArray;
00085 BOOST_FOREACH (const std::string& name, _airportCodeList) {
00086     lAirportCodeArray.push_back (std::pair<bpt::ptree::key_type,
00087                                     bpt::ptree::data_type> ("", name))
00088 ;
00089     }
00089 pt.put_child ("flight_date.airport_codes", lAirportCodeArray);
00090 //pt.push_back (std::make_pair ("flight_date.airport_codes",
00091 lAirportCodeArray));
00092
00092 // Write the property tree to the JSON stream.
00093 write_json (oStr, pt);
00094
00095 return oStr.str();
00096 }
00097
00098 }

```

25.187 airinv/server/BomPropertyTree.hpp File Reference

```

#include <string>
#include <set>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

Classes

- struct [stdair::BomPropertyTree](#)

Namespaces

- namespace [stdair](#)
Forward declarations.

25.188 BomPropertyTree.hpp

```

00001 #ifndef __AIRINV_SVR_BOMPROPERTYTREE_HPP
00002 #define __AIRINV_SVR_BOMPROPERTYTREE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <set>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013
00014 namespace stdair {
00015
00016     struct BomPropertyTree {
00017         void load (const std::string& iBomTree);
00018
00019         std::string save() const;
00020
00021         // ////////////////////////////////// Attributes //////////////////////////////////
00022         stdair::AirlineCode_T _airlineCode;
00023     };
00024 }
00025
00026 #endif

```

```

00034
00036     stdair::FlightNumber_T _flightNumber;
00037
00039     stdair::Date_T _departureDate;
00040
00042     std::set<stdair::AirportCode_T> _airportCodeList;
00043 };
00044
00045 }
00046 #endif // __AIRINV_SVR_BOMPROPERTYTREE_HPP

```

25.189 airinv/server/Connection.cpp File Reference

```

#include <cassert>
#include <vector>
#include <boost/bind.hpp>
#include <airinv/server/RequestHandler.hpp>
#include <airinv/server/Connection.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.190 Connection.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <vector>
00007 // Boost
00008 #include <boost/bind.hpp>
00009 // AirInv
00010 #include <airinv/server/RequestHandler.hpp>
00011 #include <airinv/server/Connection.hpp>
00012
00013 namespace AIRINV {
00014
00015 // //////////////////////////////////////
00016 Connection::Connection (boost::asio::io_service& ioService,
00017                        RequestHandler& ioHandler)
00018 : _strand (ioService), _socket (ioService), _requestHandler (ioHandler) {
00019 }
00020
00021 // //////////////////////////////////////
00022 boost::asio::ip::tcp::socket& Connection::socket() {
00023     return _socket;
00024 }
00025
00026 // //////////////////////////////////////
00027 void Connection::start() {
00028
00029     _socket.async_read_some (boost::asio::buffer (_buffer),
00030                             _strand.wrap (boost::bind (&Connection::handleRead
00031
00032                                     shared_from_this(),
00033
00034                                     boost::asio::placeholders::error,
00035                                     boost::asio::placeholders::bytes_transferred)));
00036 }
00037
00038 // //////////////////////////////////////
00039 void Connection::handleRead (const boost::system::error_code& iErrorCode,
00040                             std::size_t bytes_transferred) {
00041     if (!iErrorCode) {
00042         _request._flightDetails = _buffer.data();
00043         const bool hasBeenSuccessful = _requestHandler.handleRequest (_request,
00044                                     _reply);
00045
00046         if (hasBeenSuccessful == true) {
00047             boost::asio::async_write (_socket, _reply.to_buffers(),

```

```

00047         _strand.wrap (boost::bind (&
00048     Connection::handleWrite,
00049                                     shared_from_this())
00050     ,
00051     boost::asio::placeholders::error));
00052     } else {
00053         boost::asio::async_write (_socket, _reply.to_buffers(),
00054     _strand.wrap (boost::bind (&
00055     Connection::handleWrite,
00056                                     shared_from_this())
00057     ,
00058     boost::asio::placeholders::error));
00059     }
00060     // If an error occurs then no new asynchronous operations are
00061     // started. This means that all shared_ptr references to the
00062     // connection object will disappear and the object will be
00063     // destroyed automatically after this handler returns. The
00064     // connection class's destructor closes the socket.
00065     }
00066     // //////////////////////////////////////
00067     void Connection::handleWrite (const boost::system::error_code& iErrorCode) {
00068     if (!iErrorCode) {
00069         // Initiate graceful connection closure.
00070         boost::system::error_code ignored_ec;
00071         _socket.shutdown (boost::asio::ip::tcp::socket::shutdown_both,
00072             ignored_ec);
00073     }
00074     // No new asynchronous operations are started. This means that all
00075     // shared_ptr references to the connection object will disappear
00076     // and the object will be destroyed automatically after this
00077     // handler returns. The connection class's destructor closes the
00078     // socket.
00079     }
00080     }
00081     }
00082     }
00083     }
00084     }
00085     }

```

25.191 airinv/server/Connection.hpp File Reference

```

#include <boost/asio.hpp>
#include <boost/array.hpp>
#include <boost/noncopyable.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <airinv/server/Reply.hpp>
#include <airinv/server/Request.hpp>

```

Classes

- class [AIRINV::Connection](#)

Namespaces

- namespace [AIRINV](#)

Typedefs

- typedef boost::shared_ptr
< Connection > [AIRINV::ConnectionShrPtr_T](#)

25.192 Connection.hpp

```

00001 #ifndef __AIRINV_SVR_CONNECTION_HPP
00002 #define __AIRINV_SVR_CONNECTION_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 // Boost
00009 #include <boost/asio.hpp>
00010 #include <boost/array.hpp>
00011 #include <boost/noncopyable.hpp>
00012 #include <boost/shared_ptr.hpp>
00013 #include <boost/enable_shared_from_this.hpp>
00014 // AirInv
00015 #include <airinv/server/Reply.hpp>
00016 #include <airinv/server/Request.hpp>
00017
00018 namespace AIRINV {
00019
00020     // Forward declarations.
00021     class RequestHandler;
00022
00023
00024
00025     class Connection : public boost::enable_shared_from_this<Connection>,
00026                       private boost::noncopyable {
00027     public:
00028         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00031         Connection (boost::asio::io_service&, RequestHandler&);
00032
00033
00034         // ////////////////////////////////// Business Support Methods //////////////////////////////////
00036         boost::asio::ip::tcp::socket& socket();
00037
00039         void start();
00040
00041
00042     private:
00044         void handleRead (const boost::system::error_code& e,
00045                         std::size_t bytes_transferred);
00046
00048         void handleWrite (const boost::system::error_code& e);
00049
00052         boost::asio::io_service::strand _strand;
00053
00055         boost::asio::ip::tcp::socket _socket;
00056
00058         RequestHandler& _requestHandler;
00059
00061         boost::array<char, 8192> _buffer;
00062
00064         Request _request;
00065
00067         Reply _reply;
00068     };
00069
00071     typedef boost::shared_ptr<Connection> ConnectionShrPtr_T;
00072
00073 }
00074 #endif // __AIRINV_SVR_CONNECTION_HPP

```

25.193 airinv/server/header.hpp File Reference

```
#include <string>
```

Classes

- struct [AIRINV::header](#)

Namespaces

- namespace [AIRINV](#)

25.194 header.hpp

```

00001 #ifndef __AIRINV_SVR_HEADER_HPP
00002 #define __AIRINV_SVR_HEADER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace AIRINV {
00011
00012     struct header {
00013         std::string name;
00014         std::string value;
00015     };
00016
00017 }
00018
00019 #endif // __AIRINV_SVR_HEADER_HPP

```

25.195 airinv/server/posix_main.cpp File Reference

```

#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/thread.hpp>
#include <boost/bind.hpp>
#include <boost/lexical_cast.hpp>
#include <airinv/server/AirInvServer.hpp>
#include <pthread.h>
#include <signal.h>

```

Functions

- int [main](#) (int argc, char *argv[])

25.195.1 Function Documentation

25.195.1.1 int main (int argc, char * argv[])

Definition at line 25 of file [posix_main.cpp](#).

References [AIRINV::AirInvServer::run\(\)](#).

25.196 posix_main.cpp

```

00001 //
00002 // posix_main.cpp
00003 // ~~~~~
00004 //
00005 // Copyright (c) 2003-2008 Christopher M. Kohlhoff (chris at kohlhoff dot com)
00006 //
00007 // Distributed under the Boost Software License, Version 1.0. (See accompanying
00008 // file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
00009 //
00010
00011 #include <iostream>
00012 #include <string>
00013 #include <boost/asio.hpp>
00014 #include <boost/thread.hpp>
00015 #include <boost/bind.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 #include <airinv/server/AirInvServer.hpp>
00018
00019 #if !defined(_WIN32)
00020
00021 #include <pthread.h>

```

```

00022 #include <signal.h>
00023
00024 // ////////////////////////////////// M A I N //////////////////////////////////
00025 int main(int argc, char* argv[]) {
00026
00027     try {
00028
00029         // Check command line arguments.
00030         if (argc != 5) {
00031             std::cerr << "Usage: airinvServer <address> <port> <threads> <doc_root>"
00032                 << std::endl;
00033             std::cerr << "    For IPv4, try:" << std::endl;
00034             std::cerr << "        receiver 0.0.0.0 80 1 ." << std::endl;
00035             std::cerr << "    For IPv6, try:" << std::endl;
00036             std::cerr << "        receiver 0::0 80 1 ." << std::endl;
00037             return 1;
00038         }
00039
00040         // Block all signals for background thread.
00041         sigset_t new_mask;
00042         sigfillset (&new_mask);
00043         sigset_t old_mask;
00044         pthread_sigmask (SIG_BLOCK, &new_mask, &old_mask);
00045
00046         // Run server in background thread.
00047         std::size_t num_threads = boost::lexical_cast<std::size_t>(argv[3]);
00048         AIRINV::AirInvServer s (argv[1], argv[2], argv[4], num_threads);
00049         boost::thread t (boost::bind (&AIRINV::AirInvServer::run, &s));
00050
00051         // Restore previous signals.
00052         pthread_sigmask (SIG_SETMASK, &old_mask, 0);
00053
00054         // Wait for signal indicating time to shut down.
00055         sigset_t wait_mask;
00056         sigemptyset (&wait_mask);
00057         sigaddset (&wait_mask, SIGINT);
00058         sigaddset (&wait_mask, SIGQUIT);
00059         sigaddset (&wait_mask, SIGTERM);
00060         pthread_sigmask (SIG_BLOCK, &wait_mask, 0);
00061         int sig = 0;
00062         sigwait (&wait_mask, &sig);
00063
00064         // Stop the server.
00065         s.stop();
00066         t.join();
00067
00068     } catch (std::exception& e) {
00069         std::cerr << "exception: " << e.what() << "\n";
00070     }
00071
00072     return 0;
00073 }
00074
00075 #endif // !defined(_WIN32)

```

25.197 airinv/server/Reply.cpp File Reference

```

#include <cassert>
#include <string>
#include <boost/lexical_cast.hpp>
#include <airinv/server/Reply.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.198 Reply.cpp

```

00001 // //////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // Boost

```

```

00008 #include <boost/lexical_cast.hpp>
00009 // AirInv
00010 #include <airinv/server/Reply.hpp>
00011
00012 namespace AIRINV {
00013
00014 // //////////////////////////////////////
00015 std::vector<boost::asio::const_buffer> Reply::to_buffers() {
00016     std::vector<boost::asio::const_buffer> lBuffers;
00017     lBuffers.push_back (boost::asio::buffer(content));
00018     return lBuffers;
00019 }
00020
00021 }

```

25.199 airinv/server/Reply.hpp File Reference

```

#include <string>
#include <vector>
#include <boost/asio.hpp>
#include <airinv/FlightRequestStatus.hpp>

```

Classes

- struct [AIRINV::Reply](#)

Namespaces

- namespace [AIRINV](#)

25.200 Reply.hpp

```

00001 #ifndef __AIRINV_SVR_REPLY_HPP
00002 #define __AIRINV_SVR_REPLY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // Boost
00011 #include <boost/asio.hpp>
00012 // AirInv
00013 #include <airinv/FlightRequestStatus.hpp>
00014
00015 namespace AIRINV {
00016
00017     struct Reply {
00020         FlightRequestStatus::EN_FlightRequestStatus _status;
00021
00023         std::string content;
00024
00029         std::vector<boost::asio::const_buffer> to_buffers();
00030     };
00031
00032 }
00033 #endif // __AIRINV_SVR_REPLY_HPP

```

25.201 airinv/server/Request.cpp File Reference

```

#include <cassert>
#include <airinv/server/Request.hpp>

```


Namespaces

- namespace [AIRINV](#)

25.202 Request.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AirInv
00007 #include <airinv/server/Request.hpp>
00008
00009 namespace AIRINV {
00010
00011 // //////////////////////////////////////
00012 bool Request::parseFlightDate () {
00013     bool hasBeenSuccessfull = false;
00014
00015     //
00016     _airlineCode = "BA";
00017     _flightNumber = 341;
00018     _departureDate = stdair::Date_T (2010, 04, 20);
00019
00020     //
00021     hasBeenSuccessfull = true;
00022
00023     return hasBeenSuccessfull;
00024 }
00025
00026 }
```

25.203 airinv/server/Request.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct [AIRINV::Request](#)

Namespaces

- namespace [AIRINV](#)

25.204 Request.hpp

```

00001 #ifndef __AIRINV_SVR_REQUEST_HPP
00002 #define __AIRINV_SVR_REQUEST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 // AirInv
00014
00015 namespace AIRINV {
00016
00017     struct Request {
00018     public:
```

```

00024     bool parseFlightDate();
00025
00026 public:
00027     // ////////// Attributes //////////
00029     std::string _flightDetails;
00031     stdair::AirlineCode_T _airlineCode;
00033     stdair::FlightNumber_T _flightNumber;
00035     stdair::Date_T _departureDate;
00036 };
00037
00038 }
00039 #endif // __AIRINV_SVR_REQUEST_HPP

```

25.205 airinv/server/RequestHandler.cpp File Reference

```

#include <cassert>
#include <string>
#include <fstream>
#include <sstream>
#include <boost/lexical_cast.hpp>
#include <airinv/server/Reply.hpp>
#include <airinv/server/Request.hpp>
#include <airinv/server/RequestHandler.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.206 RequestHandler.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <fstream>
00008 #include <sstream>
00009 // Boost
00010 #include <boost/lexical_cast.hpp>
00011 // StdAir
00012 // AirInv
00013 #include <airinv/server/Reply.hpp>
00014 #include <airinv/server/Request.hpp>
00015 #include <airinv/server/RequestHandler.hpp>
00016
00017 namespace AIRINV {
00018
00019     // //////////////////////////////////////
00020     RequestHandler::RequestHandler (const stdair::AirlineCode_T& iAirlineCode)
00021         : _airlineCode (iAirlineCode) {
00022     }
00023
00024     // //////////////////////////////////////
00025     bool RequestHandler::
00026     handleRequest (Request& ioRequest, Reply& ioReply) const {
00027         bool hasBeenSuccessfull = false;
00028
00029         // Decode request string to a flight-date details (airline code,
00030         // flight number and departure date)
00031         hasBeenSuccessfull = ioRequest.parseFlightDate();
00032
00033         if (hasBeenSuccessfull == false) {
00034             ioReply._status = FlightRequestStatus::INTERNAL_ERROR;
00035             return hasBeenSuccessfull;
00036         }
00037
00045         // Fill out the reply to be sent to the client.
00046         ioReply._status = FlightRequestStatus::OK;
00047         ioReply.content = "Your are looking for: '" + ioRequest._flightDetails + "
'. Ok, I have found your flight-date. Be patient until I give you more
information about it";
00048

```

```

00049     return hasBeenSuccessful;
00050 }
00051
00052 }
```

25.207 airinv/server/RequestHandler.hpp File Reference

```

#include <string>
#include <boost/noncopyable.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class [AIRINV::RequestHandler](#)
The common handler for all incoming requests.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRINV](#)

25.208 RequestHandler.hpp

```

00001 #ifndef __AIRINV_SVR_REQUESTHANDLER_HPP
00002 #define __AIRINV_SVR_REQUESTHANDLER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/noncopyable.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 // AirInv
00014
00015 // Forward declarations
00016 namespace stdair {
00017     struct InventoryKey_T;
00018     struct FlightDateKey_T;
00019 }
00020
00021 namespace AIRINV {
00022
00023     // Forward declarations.
00024     struct Reply;
00025     struct Request;
00026
00027     class RequestHandler : private boost::noncopyable {
00028     public:
00029         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00030         RequestHandler (const stdair::AirlineCode_T&);
00031
00032     public:
00033         // ////////////////////////////////// Business Support Methods //////////////////////////////////
00034         bool handleRequest (Request&, Reply&) const;
00035
00036     private:
00037         // ////////////////////////////////// Attributes //////////////////////////////////
00038         stdair::AirlineCode_T _airlineCode;
00039     };
00040 }
00041 #endif // __AIRINV_SVR_REQUESTHANDLER_HPP
```

25.209 airinv/server/RequestParser.cpp File Reference

```
#include <cassert>
#include <airinv/server/RequestParser.hpp>
#include <airinv/server/Request.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.210 RequestParser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AirInv
00007 #include <airinv/server/RequestParser.hpp>
00008 #include <airinv/server/Request.hpp>
00009
00010 namespace AIRINV {
00011
00012 // //////////////////////////////////////
00013 RequestParser::RequestParser()
00014 : state_(method_start) {
00015 }
00016
00017 // //////////////////////////////////////
00018 void RequestParser::reset() {
00019     state_ = method_start;
00020 }
00021
00022 // //////////////////////////////////////
00023 boost::tribool RequestParser::consume (Request& req, char input) {
00024
00025     switch (state_) {
00026
00027     case method_start:
00028         if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00029             return false;
00030         } else {
00031             state_ = method;
00032             req.method.push_back(input);
00033             return boost::indeterminate;
00034         }
00035
00036     case method:
00037         if (input == ' ') {
00038             state_ = uri;
00039             return boost::indeterminate;
00040         } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00041             return false;
00042         } else {
00043             req.method.push_back(input);
00044             return boost::indeterminate;
00045         }
00046
00047     case uri_start:
00048         if (is_ctl(input)) {
00049             return false;
00050         } else {
00051             state_ = uri;
00052             req.uri.push_back(input);
00053             return boost::indeterminate;
00054         }
00055
00056     case uri:
00057         if (input == ' ') {
00058             state_ = http_version_h;
00059             return boost::indeterminate;
00060         } else if (is_ctl(input)) {
00061             return false;
00062         }
```

```
00067
00068     } else {
00069         req.uri.push_back(input);
00070         return boost::indeterminate;
00071     }
00072
00073     case http_version_h:
00074         if (input == 'H') {
00075             state_ = http_version_t_1;
00076             return boost::indeterminate;
00077         } else {
00078             return false;
00079         }
00080     }
00081
00082     case http_version_t_1:
00083         if (input == 'T') {
00084             state_ = http_version_t_2;
00085             return boost::indeterminate;
00086         } else {
00087             return false;
00088         }
00089     }
00090
00091     case http_version_t_2:
00092         if (input == 'T') {
00093             state_ = http_version_p;
00094             return boost::indeterminate;
00095         } else {
00096             return false;
00097         }
00098     }
00099
00100     case http_version_p:
00101         if (input == 'P') {
00102             state_ = http_version_slash;
00103             return boost::indeterminate;
00104         } else {
00105             return false;
00106         }
00107     }
00108
00109     case http_version_slash:
00110         if (input == '/') {
00111             req.http_version_major = 0;
00112             req.http_version_minor = 0;
00113             state_ = http_version_major_start;
00114             return boost::indeterminate;
00115         } else {
00116             return false;
00117         }
00118     }
00119
00120     case http_version_major_start:
00121         if (is_digit(input)) {
00122             req.http_version_major = req.http_version_major * 10 + input - '0';
00123             state_ = http_version_major;
00124             return boost::indeterminate;
00125         } else {
00126             return false;
00127         }
00128     }
00129
00130     case http_version_major:
00131         if (input == '.') {
00132             state_ = http_version_minor_start;
00133             return boost::indeterminate;
00134         } else if (is_digit(input)) {
00135             req.http_version_major = req.http_version_major * 10 + input - '0';
00136             return boost::indeterminate;
00137         } else {
00138             return false;
00139         }
00140     }
00141
00142     case http_version_minor_start:
00143         if (is_digit(input)) {
00144             req.http_version_minor = req.http_version_minor * 10 + input - '0';
00145             state_ = http_version_minor;
00146             return boost::indeterminate;
00147         } else {
00148             return false;
00149         }
00150     }
00151
00152     case http_version_minor:
```

```
00154     if (input == '\r') {
00155         state_ = expecting_newline_1;
00156         return boost::indeterminate;
00157     } else if (is_digit(input)) {
00158         req.http_version_minor = req.http_version_minor * 10 + input - '0';
00159         return boost::indeterminate;
00160     } else {
00161         return false;
00162     }
00163 }
00164
00165 case expecting_newline_1:
00166     if (input == '\n') {
00167         state_ = header_line_start;
00168         return boost::indeterminate;
00169     } else {
00170         return false;
00171     }
00172 }
00173
00174 case header_line_start:
00175     if (input == '\r') {
00176         state_ = expecting_newline_3;
00177         return boost::indeterminate;
00178     } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00179         return false;
00180     } else {
00181         state_ = header_name;
00182         return boost::indeterminate;
00183     }
00184 }
00185
00186 case header_lws:
00187     if (input == '\r') {
00188         state_ = expecting_newline_2;
00189         return boost::indeterminate;
00190     } else if (input == ' ' || input == '\t') {
00191         return boost::indeterminate;
00192     } else if (is_ctl(input)) {
00193         return false;
00194     } else {
00195         state_ = header_value;
00196         return boost::indeterminate;
00197     }
00198 }
00199
00200 case header_name:
00201     if (input == ':') {
00202         state_ = space_before_header_value;
00203         return boost::indeterminate;
00204     } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00205         return false;
00206     } else {
00207         return boost::indeterminate;
00208     }
00209 }
00210
00211 case space_before_header_value:
00212     if (input == ' ') {
00213         state_ = header_value;
00214         return boost::indeterminate;
00215     } else {
00216         return false;
00217     }
00218 }
00219
00220 case header_value:
00221     if (input == '\r') {
00222         state_ = expecting_newline_2;
00223         return boost::indeterminate;
00224     } else if (is_ctl(input)) {
00225         return false;
00226     } else {
00227         return boost::indeterminate;
00228     }
00229 }
00230
00231 case expecting_newline_2:
00232     if (input == '\n') {
00233         state_ = header_line_start;
00234         return boost::indeterminate;
00235     }
00236 }
```

```

00241
00242     } else {
00243         return false;
00244     }
00245
00246     case expecting_newline_3:
00247         return (input == '\n');
00248
00249     default:
00250         return false;
00251     }
00252 }
00253
00254 ///////////////////////////////////////////////////////////////////
00255 bool RequestParser::is_char(int c) {
00256     return c >= 0 && c <= 127;
00257 }
00258
00259 ///////////////////////////////////////////////////////////////////
00260 bool RequestParser::is_ctl(int c) {
00261     return (c >= 0 && c <= 31) || (c == 127);
00262 }
00263
00264 ///////////////////////////////////////////////////////////////////
00265 bool RequestParser::is_tspecial(int c) {
00266     switch (c) {
00267         case '(': case ')': case '<': case '>': case '@':
00268         case ',': case ';': case ':': case '\\': case '"':
00269         case '/': case '[': case ']': case '?': case '=':
00270         case '{': case '}': case ' ': case '\t':
00271             return true;
00272         default:
00273             return false;
00274     }
00275 }
00276
00277 ///////////////////////////////////////////////////////////////////
00278 bool RequestParser::is_digit(int c) {
00279     return c >= '0' && c <= '9';
00280 }
00281
00282 }

```

25.211 airinv/server/RequestParser.hpp File Reference

```

#include <boost/logic/tribool.hpp>
#include <boost/tuple/tuple.hpp>

```

Classes

- class [AIRINV::RequestParser](#)
Parser for incoming requests.

Namespaces

- namespace [AIRINV](#)

25.212 RequestParser.hpp

```

00001 #ifndef __AIRINV_SVR_REQUESTPARSER_HPP
00002 #define __AIRINV_SVR_REQUESTPARSER_HPP
00003
00004 ///////////////////////////////////////////////////////////////////
00005 // Import section
00006 ///////////////////////////////////////////////////////////////////
00007 // STL
00008 // Boost
00009 #include <boost/logic/tribool.hpp>
00010 #include <boost/tuple/tuple.hpp>
00011
00012 namespace AIRINV {
00013

```

```

00014     struct Request;
00015
00017     class RequestParser {
00018     public:
00020         RequestParser();
00021
00023         void reset();
00024
00029         template <typename InputIterator>
00030         boost::tuple<boost::tribool, InputIterator> parse (Request& req,
00031                                                         InputIterator begin,
00032                                                         InputIterator end) {
00033
00034             while (begin != end) {
00035                 boost::tribool result = consume(req, *begin++);
00036                 if (result || !result)
00037                     return boost::make_tuple(result, begin);
00038             }
00039
00040             boost::tribool result = boost::indeterminate;
00041             return boost::make_tuple(result, begin);
00042         }
00043
00044     private:
00046         boost::tribool consume (Request& req, char input);
00047
00049         static bool is_char(int c);
00050
00052         static bool is_ctl(int c);
00053
00055         static bool is_tspecial(int c);
00056
00058         static bool is_digit(int c);
00059
00061         enum state {
00062             method_start,
00063             method,
00064             uri_start,
00065             uri,
00066             http_version_h,
00067             http_version_t_1,
00068             http_version_t_2,
00069             http_version_p,
00070             http_version_slash,
00071             http_version_major_start,
00072             http_version_major,
00073             http_version_minor_start,
00074             http_version_minor,
00075             expecting_newline_1,
00076             header_line_start,
00077             header_lws,
00078             header_name,
00079             space_before_header_value,
00080             header_value,
00081             expecting_newline_2,
00082             expecting_newline_3
00083         } state_;
00084     };
00085
00086 }
00087 #endif // __AIRINV_SVR_REQUESTPARSER_HPP

```

25.213 airinv/server/win_main.cpp File Reference

```

#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/lexical_cast.hpp>
#include <airinv/server/AirInvServer.hpp>

```

25.214 win_main.cpp

```

00001 //
00002 // win_main.cpp

```



```

00003 // ~~~~~
00004 //
00005 // Copyright (c) 2003-2008 Christopher M. Kohlhoff (chris at kohlhoff dot com)
00006 //
00007 // Distributed under the Boost Software License, Version 1.0. (See accompanying
00008 // file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
00009 //
00010
00011 #include <iostream>
00012 #include <string>
00013 #include <boost/asio.hpp>
00014 #include <boost/bind.hpp>
00015 #include <boost/function.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 #include <airinv/server/AirInvServer.hpp>
00018
00019 #if defined(_WIN32)
00020
00021 boost::function0<void> console_ctrl_function;
00022
00023 BOOL WINAPI console_ctrl_handler(DWORD ctrl_type) {
00024     switch (ctrl_type) {
00025     case CTRL_C_EVENT:
00026     case CTRL_BREAK_EVENT:
00027     case CTRL_CLOSE_EVENT:
00028     case CTRL_SHUTDOWN_EVENT:
00029         console_ctrl_function();
00030         return TRUE;
00031     default:
00032         return FALSE;
00033     }
00034 }
00035
00036 int main(int argc, char* argv[]) {
00037     try {
00038         // Check command line arguments.
00039         if (argc != 5) {
00040             std::cerr << "Usage: http_server <address> <port> <threads> <doc_root>\n"
00041 ;
00042             std::cerr << "  For IPv4, try:\n";
00043             std::cerr << "    http_server 0.0.0.0 80 1 .\n";
00044             std::cerr << "  For IPv6, try:\n";
00045             std::cerr << "    http_server 0::0 80 1 .\n";
00046             return 1;
00047         }
00048     }
00049
00050     // Initialise server.
00051     std::size_t num_threads = boost::lexical_cast<std::size_t>(argv[3]);
00052     AIRINV::AirInvServer s (argv[1], argv[2], argv[4], num_threads);
00053
00054     // Set console control handler to allow server to be stopped.
00055     console_ctrl_function = boost::bind(&AIRINV::AirInvServer::stop, &s);
00056     SetConsoleCtrlHandler(console_ctrl_handler, TRUE);
00057
00058     // Run the server until stopped.
00059     s.run();
00060
00061     catch (std::exception& e) {
00062         std::cerr << "exception: " << e.what() << "\n";
00063     }
00064
00065     return 0;
00066 }
00067 #endif // defined(_WIN32)

```

25.215 airinv/service/AIRINV_Master_Service.cpp File Reference

```
#include <cassert>
```

```

#include <cmath>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/EventQueue.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/factory/FacAirinvMasterServiceContext.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
#include <airinv/AIRINV_Service.hpp>
#include <airinv/AIRINV_Master_Service.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.216 AIRINV_Master_Service.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <cmath>
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/basic/BasChronometer.hpp>
00011 #include <stdair/basic/EventType.hpp>
00012 #include <stdair/bom/BomKeyManager.hpp>
00013 #include <stdair/bom/EventQueue.hpp>
00014 #include <stdair/bom/SnapshotStruct.hpp>
00015 #include <stdair/bom/RMEventStruct.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 #include <stdair/STDAIR_Service.hpp>
00018 // AirInv
00019 #include <airinv/basic/BasConst_AIRINV_Service.hpp>
00020 #include <airinv/factory/FacAirinvMasterServiceContext.hpp>
00021 #include <airinv/command/InventoryParser.hpp>
00022 #include <airinv/command/InventoryManager.hpp>
00023 #include <airinv/service/AIRINV_Master_ServiceContext.hpp>
00024 #include <airinv/AIRINV_Service.hpp>
00025 #include <airinv/AIRINV_Master_Service.hpp>
00026
00027 namespace AIRINV {
00028
00029 // //////////////////////////////////////
00030 AIRINV_Master_Service::AIRINV_Master_Service()
00031 : _airinvMasterServiceContext (NULL) {
00032     assert (false);
00033 }
00034
00035 // //////////////////////////////////////
00036 AIRINV_Master_Service::
00037 AIRINV_Master_Service (const AIRINV_Master_Service& iService)
00038 : _airinvMasterServiceContext (NULL) {
00039     assert (false);
00040 }
00041
00042 // //////////////////////////////////////
00043 AIRINV_Master_Service::
00044 AIRINV_Master_Service (const stdair::BasLogParams& iLogParams,
00045                       const stdair::BasDBParams& iDBParams)
00046 : _airinvMasterServiceContext (NULL) {
00047
00048     // Initialise the STDAIR service handler

```

```

00049     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00050         initStdAirService (iLogParams, iDBParams);
00051
00052     // Initialise the service context
00053     initServiceContext();
00054
00055     // Add the StdAir service context to the AIRINV service context
00056     // \note RMOL owns the STDAIR service resources here.
00057     const bool ownStdairService = true;
00058     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00059
00060     // Initialise the (remaining of the) context
00061     initSlaveAirinvService();
00062 }
00063
00064 // //////////////////////////////////////
00065 AIRINV_Master_Service::
00066 AIRINV_Master_Service (const stdair::BasLogParams& iLogParams)
00067     : _airinvMasterServiceContext (NULL) {
00068
00069     // Initialise the STDAIR service handler
00070     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00071         initStdAirService (iLogParams);
00072
00073     // Initialise the service context
00074     initServiceContext();
00075
00076     // Add the StdAir service context to the AIRINV service context
00077     // \note RMOL owns the STDAIR service resources here.
00078     const bool ownStdairService = true;
00079     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00080
00081     // Initialise the (remaining of the) context
00082     initSlaveAirinvService();
00083 }
00084
00085 // //////////////////////////////////////
00086 AIRINV_Master_Service::
00087 AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00088     : _airinvMasterServiceContext (NULL) {
00089
00090     // Initialise the service context
00091     initServiceContext();
00092
00093     // Store the STDAIR service object within the (AIRINV) service context
00094     // \note AirInv does not own the STDAIR service resources here.
00095     const bool doesNotOwnStdairService = false;
00096     addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00097
00098     // Initialise the (remaining of the) context
00099     initSlaveAirinvService();
00100 }
00101
00102 // //////////////////////////////////////
00103 AIRINV_Master_Service::~AIRINV_Master_Service() {
00104     // Delete/Clean all the objects from memory
00105     finalise();
00106 }
00107
00108 // //////////////////////////////////////
00109 void AIRINV_Master_Service::finalise() {
00110     assert (_airinvMasterServiceContext != NULL);
00111     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00112     _airinvMasterServiceContext->reset();
00113 }
00114
00115 // //////////////////////////////////////
00116 void AIRINV_Master_Service::initServiceContext() {
00117     // Initialise the context
00118     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00119         FacAirinvMasterServiceContext::instance().create();
00120     _airinvMasterServiceContext = &lAIRINV_Master_ServiceContext;
00121 }
00122
00123 // //////////////////////////////////////
00124 void AIRINV_Master_Service::
00125 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00126                 const bool iOwnStdairService) {
00127
00128     // Retrieve the AirInv Master service context
00129     assert (_airinvMasterServiceContext != NULL);
00130     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00131         *_airinvMasterServiceContext;
00132
00133     // Store the STDAIR service object within the (AIRINV) service context
00134     lAIRINV_Master_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00135                                                     iOwnStdairService);

```

```

00136     }
00137
00138     // //////////////////////////////////////
00139     stdair::STDAIR_ServicePtr_T AIRINV_Master_Service::
00140     initStdAirService (const stdair::BasLogParams& iLogParams,
00141                       const stdair::BasDBParams& iDBParams) {
00142
00143         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00144             boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00145
00146         return lSTDAIR_Service_ptr;
00147     }
00148
00149     // //////////////////////////////////////
00150     stdair::STDAIR_ServicePtr_T AIRINV_Master_Service::
00151     initStdAirService (const stdair::BasLogParams& iLogParams) {
00152
00153         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00154             boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00155
00156         return lSTDAIR_Service_ptr;
00157     }
00158
00159     // //////////////////////////////////////
00160     void AIRINV_Master_Service::initSlaveAirinvService() {
00161
00162         // Retrieve the AirInv Master service context
00163         assert (_airinvMasterServiceContext != NULL);
00164         AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00165             *_airinvMasterServiceContext;
00166
00167         // Retrieve the StdAir service
00168         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00169             lAIRINV_Master_ServiceContext.getSTDAIR_ServicePtr();
00170         assert (lSTDAIR_Service_ptr != NULL);
00171
00172         AIRINV_ServicePtr_T lAIRINV_Service_ptr =
00173             boost::make_shared<AIRINV_Service> (lSTDAIR_Service_ptr);
00174
00175         // Store the AIRINV service object within the AIRINV Master service
00176         context.
00177         lAIRINV_Master_ServiceContext.setAIRINV_Service (lAIRINV_Service_ptr);
00178     }
00179
00180     // //////////////////////////////////////
00181     void AIRINV_Master_Service::
00182     parseAndLoad (const stdair::Filename_T& iInventoryInputFilename) {
00183
00184         // Retrieve the AirInv Master service context
00185         if (_airinvMasterServiceContext == NULL) {
00186             throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00187                                                         "has not been initialised");
00188         }
00189
00190         assert (_airinvMasterServiceContext != NULL);
00191
00192         AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00193             *_airinvMasterServiceContext;
00194
00195         // Retrieve the slave AIRINV service object from the (AIRINV)
00196         // service context
00197         AIRINV_Service& lAIRINV_Service =
00198             lAIRINV_Master_ServiceContext.getAIRINV_Service();
00199
00200         // Delegate the file parsing and BOM building to the dedicated service
00201         lAIRINV_Service.parseAndLoad (iInventoryInputFilename);
00202     }
00203
00204     // //////////////////////////////////////
00205     void AIRINV_Master_Service::
00206     parseAndLoad (const stdair::Filename_T& iScheduleInputFilename,
00207                  const stdair::Filename_T& iODInputFilename,
00208                  const AIRRAC::YieldFilePath& iYieldFilename) {
00209
00210         // Retrieve the AirInv Master service context
00211         if (_airinvMasterServiceContext == NULL) {
00212             throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00213                                                         "has not been initialised");
00214         }
00215
00216         assert (_airinvMasterServiceContext != NULL);
00217
00218         AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00219             *_airinvMasterServiceContext;
00220
00221         // Retrieve the slave AirInv service object from the (AirInv)
00222         // service context

```

```

00243     AIRINV_Service& lAIRINV_Service =
00244         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00245
00246     // Delegate the file parsing and BOM building to the dedicated service
00247     lAIRINV_Service.parseAndLoad (iScheduleInputFilename, iODInputFilename,
00248         iYieldFilename);
00249 }
00250
00251 // //////////////////////////////////////
00252 void AIRINV_Master_Service::buildSampleBom() {
00253
00254     // Retrieve the AirInv Master service context
00255     if (_airinvMasterServiceContext == NULL) {
00256         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00257             "has not been initialised")
;
00258     }
00259     assert (_airinvMasterServiceContext != NULL);
00260
00261     // Retrieve the AirInv service context and whether it owns the Stdair
00262     // service
00263     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00264         *_airinvMasterServiceContext;
00265     const bool doesOwnStdairService =
00266         lAIRINV_Master_ServiceContext.getOwnStdairServiceFlag();
00267
00268     // Retrieve the StdAIR service object from the (AirInv) service context
00269     stdair::STDAIR_Service& lSTDAIR_Service =
00270         lAIRINV_Master_ServiceContext.getSTDAIR_Service();
00271
00272     if (doesOwnStdairService == true) {
00273         //
00274         lSTDAIR_Service.buildSampleBom();
00275     }
00276
00277     AIRINV_Service& lAIRINV_Service =
00278         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00279     lAIRINV_Service.buildSampleBom();
00280 }
00281
00282 // //////////////////////////////////////
00283 std::string AIRINV_Master_Service::
00284 jsonExport (const stdair::AirlineCode_T& iAirlineCode,
00285     const stdair::FlightNumber_T& iFlightNumber,
00286     const stdair::Date_T& iDepartureDate) const {
00287
00288     // Retrieve the AirInv Master service context
00289     if (_airinvMasterServiceContext == NULL) {
00290         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00291             "has not been initialised")
;
00292     }
00293     assert (_airinvMasterServiceContext != NULL);
00294
00295     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00296         *_airinvMasterServiceContext;
00297
00298     // Retrieve the slave AirInv (slave) service object from
00299     // the (AirInv master) service context
00300     AIRINV_Service& lAIRINV_Service =
00301         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00302
00303     // Delegate the BOM dump to the dedicated service
00304     return lAIRINV_Service.jsonExport (iAirlineCode, iFlightNumber,
00305         iDepartureDate);
00306 }
00307
00308 // //////////////////////////////////////
00309 std::string AIRINV_Master_Service::
00310 list (const stdair::AirlineCode_T& iAirlineCode,
00311     const stdair::FlightNumber_T& iFlightNumber) const {
00312     std::ostringstream oFlightListStr;
00313
00314     // Retrieve the AirInv Master service context
00315     if (_airinvMasterServiceContext == NULL) {
00316         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00317             "has not been initialised")
;
00318     }
00319     assert (_airinvMasterServiceContext != NULL);
00320
00321     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00322         *_airinvMasterServiceContext;
00323
00324     // Retrieve the slave AirInv (slave) service object from
00325     // the (AirInv master) service context

```

```

00346     AIRINV_Service& lAIRINV_Service =
00347         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00348
00349     // Delegate the BOM display to the dedicated service
00350     return lAIRINV_Service.list (iAirlineCode, iFlightNumber);
00351 }
00352
00353 // //////////////////////////////////////
00354 bool AIRINV_Master_Service::
00355 check (const stdair::AirlineCode_T& iAirlineCode,
00356        const stdair::FlightNumber_T& iFlightNumber,
00357        const stdair::Date_T& iDepartureDate) const {
00358     std::ostringstream oFlightListStr;
00359
00360     // Retrieve the AirInv Master service context
00361     if (_airinvMasterServiceContext == NULL) {
00362         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00363             "has not been initialised")
00364     ;
00365     }
00366     assert (_airinvMasterServiceContext != NULL);
00367
00368     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00369         *_airinvMasterServiceContext;
00370
00371     // Retrieve the slave AirInv (slave) service object from
00372     // the (AirInv master) service context
00373     AIRINV_Service& lAIRINV_Service =
00374         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00375
00376     // Delegate the BOM display to the dedicated service
00377     return lAIRINV_Service.check (iAirlineCode, iFlightNumber, iDepartureDate);
00378 }
00379 // //////////////////////////////////////
00380 std::string AIRINV_Master_Service::csvDisplay() const {
00381
00382     // Retrieve the AirInv Master service context
00383     if (_airinvMasterServiceContext == NULL) {
00384         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00385             "has not been initialised")
00386     ;
00387     }
00388     assert (_airinvMasterServiceContext != NULL);
00389
00390     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00391         *_airinvMasterServiceContext;
00392
00393     // Retrieve the slave AIRINV service object from
00394     // the (AIRINV) service context
00395     AIRINV_Service& lAIRINV_Service =
00396         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00397
00398     // Delegate the BOM display to the dedicated service
00399     return lAIRINV_Service.csvDisplay();
00400 }
00401 // //////////////////////////////////////
00402 std::string AIRINV_Master_Service::
00403 csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00404            const stdair::FlightNumber_T& iFlightNumber,
00405            const stdair::Date_T& iDepartureDate) const {
00406
00407     // Retrieve the AirInv Master service context
00408     if (_airinvMasterServiceContext == NULL) {
00409         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00410             "has not been initialised")
00411     ;
00412     }
00413     assert (_airinvMasterServiceContext != NULL);
00414
00415     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00416         *_airinvMasterServiceContext;
00417
00418     // Retrieve the slave AIRINV service object from
00419     // the (AIRINV) service context
00420     AIRINV_Service& lAIRINV_Service =
00421         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00422
00423     // Delegate the BOM display to the dedicated service
00424     return lAIRINV_Service.csvDisplay (iAirlineCode, iFlightNumber,
00425         iDepartureDate);
00426 }
00427 // //////////////////////////////////////
00428 void AIRINV_Master_Service::
00429 initSnapshotAndRMEvents (const stdair::Date_T& iStartDate,

```

```

00430         const stdair::Date_T& iEndDate) {
00431
00432     // Retrieve the AirInv Master service context
00433     if (_airinvMasterServiceContext == NULL) {
00434         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00435             "has not been initialised")
;
00436     }
00437     assert (_airinvMasterServiceContext != NULL);
00438
00439     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00440         *_airinvMasterServiceContext;
00441
00442     // Retrieve the StdAIR service context
00443     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00444         lAIRINV_Master_ServiceContext.getSTDAIR_ServicePtr();
00445     assert (lSTDAIR_Service_ptr != NULL);
00446
00447     // Retrieve the event queue object instance
00448     stdair::EventQueue& lQueue = lSTDAIR_Service_ptr->getEventQueue();
00449
00450     // Initialise the snapshot events
00451     InventoryManager::initSnapshotEvents (iStartDate, iEndDate, lQueue);
00452
00453     // \todo Browse the list of inventories and itinitialise the RM events of
00454     // each inventory.
00455
00456     // Retrieve the slave AIRINV service object from the (AIRINV)
00457     // service context
00458     AIRINV_Service& lAIRINV_Service =
00459         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00460     lQueue.addStatus (stdair::EventType::RM, 0);
00461     stdair::RMEventList_T lRMEventList =
00462         lAIRINV_Service.initRMEvents (iStartDate, iEndDate);
00463     InventoryManager::addRMEventsToEventQueue (lQueue, lRMEventList);
00464 }
00465
00466 // ////////////////////////////////////////
00467 void AIRINV_Master_Service::
00468 calculateAvailability (stdair::TravelSolutionStruct& ioTravelSolution,
00469     const stdair::PartnershipTechnique&
iPartnershipTechnique) {
00470
00471     // Retrieve the AirInv Master service context
00472     if (_airinvMasterServiceContext == NULL) {
00473         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00474             "has not been initialised")
;
00475     }
00476     assert (_airinvMasterServiceContext != NULL);
00477
00478     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00479         *_airinvMasterServiceContext;
00480
00481     // Retrieve the slave AIRINV service object from the (AIRINV)
00482     // service context
00483     AIRINV_Service& lAIRINV_Service =
00484         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00485
00486     // Delegate the availability retrieval to the dedicated service
00487     stdair::BasChronometer lAvlChronometer;
00488     lAvlChronometer.start();
00489
00490     lAIRINV_Service.calculateAvailability (ioTravelSolution,
iPartnershipTechnique);
00491
00492     // DEBUG
00493     // const double lAvlMeasure = lAvlChronometer.elapsed();
00494     // STDAIR_LOG_DEBUG ("Availability retrieval: " << lAvlMeasure << " - "
00495     // << lAIRINV_Master_ServiceContext.display());
00496 }
00497
00498 // ////////////////////////////////////////
00499 bool AIRINV_Master_Service::sell (const std::string& iSegmentDateKey,
00500     const stdair::ClassCode_T& iClassCode,
00501     const stdair::PartySize_T& iPartySize) {
00502
00503     // Retrieve the AirInv Master service context
00504     if (_airinvMasterServiceContext == NULL) {
00505         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00506             "has not been initialised")
;
00507     }
00508     assert (_airinvMasterServiceContext != NULL);
00509
00510     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00511         *_airinvMasterServiceContext;

```

```

00512
00513 // Retrieve the corresponding inventory key
00514 // const stdair::InventoryKey& lInventoryKey =
00515 // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00516
00517 // Retrieve the slave AirInv service object from the (AirInv Master)
00518 // service context
00519 AIRINV_Service& lAIRINV_Service =
00520     lAIRINV_Master_ServiceContext.getAIRINV_Service();
00521
00522 // Delegate the booking to the dedicated command
00523 stdair::BasChronometer lSellChronometer;
00524 lSellChronometer.start();
00525
00526 // Delegate the BOM building to the dedicated service
00527 const bool hasBeenSaleSuccessful =
00528     lAIRINV_Service.sell (iSegmentDateKey, iClassCode, iPartySize);
00529
00530 // const double lSellMeasure = lSellChronometer.elapsed();
00531
00532 // DEBUG
00533 // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
00534 //                  << lAIRINV_Master_ServiceContext.display());
00535
00536 //
00537 return hasBeenSaleSuccessful;
00538 }
00539
00540 // //////////////////////////////////////
00541 bool AIRINV_Master_Service::cancel (const std::string& iSegmentDateKey,
00542                                     const stdair::ClassCode_T& iClassCode,
00543                                     const stdair::PartySize_T& iPartySize) {
00544
00545     // Retrieve the AirInv Master service context
00546     if (_airinvMasterServiceContext == NULL) {
00547         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00548                                                         "has not been initialised")
00549     ;
00550     }
00551     assert (_airinvMasterServiceContext != NULL);
00552
00553     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00554         *_airinvMasterServiceContext;
00555
00556     // Retrieve the corresponding inventory key
00557     // const stdair::InventoryKey& lInventoryKey =
00558     // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00559
00560     // Retrieve the slave AirInv service object from the (AirInv Master)
00561     // service context
00562     AIRINV_Service& lAIRINV_Service =
00563         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00564
00565     // Delegate the booking to the dedicated command
00566     stdair::BasChronometer lCancelChronometer;
00567     lCancelChronometer.start();
00568
00569     // Delegate the BOM building to the dedicated service
00570     const bool hasBeenSaleSuccessful =
00571         lAIRINV_Service.cancel (iSegmentDateKey, iClassCode, iPartySize);
00572
00573     // const double lCancelMeasure = lCancelChronometer.elapsed();
00574
00575     // DEBUG
00576     // STDAIR_LOG_DEBUG ("Booking cancel: " << lCancelMeasure << " - "
00577     //                  << lAIRINV_Master_ServiceContext.display());
00578
00579     //
00580     return hasBeenSaleSuccessful;
00581 }
00582
00583 // //////////////////////////////////////
00584 void AIRINV_Master_Service::
00585     takeSnapshots (const stdair::SnapshotStruct& iSnapshot) {
00586
00587     // Retrieve the AirInv Master service context
00588     if (_airinvMasterServiceContext == NULL) {
00589         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00590                                                         "has not been initialised")
00591     ;
00592     }
00593     assert (_airinvMasterServiceContext != NULL);
00594
00595     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00596         *_airinvMasterServiceContext;
00597
00598     // Retrieve the slave AIRINV service object from the (AIRINV)

```



```

00597 // service context
00598 AIRINV_Service& lAIRINV_Service =
00599     lAIRINV_Master_ServiceContext.getAIRINV_Service();
00600
00601 // Retrieve the snapshot time and the airline code.
00602 const stdair::DateTime_T& lSnapshotTime = iSnapshot.getSnapshotTime();
00603 const stdair::AirlineCode_T& lAirlineCode = iSnapshot.getAirlineCode();
00604
00605 lAIRINV_Service.takeSnapshots (lAirlineCode, lSnapshotTime);
00606 }
00607
00608 // //////////////////////////////////////
00609 void AIRINV_Master_Service::
00610 optimise (const stdair::RMEventStruct& iRMEvent,
00611          const stdair::ForecastingMethod& iForecastingMethod,
00612          const stdair::PartnershipTechnique& iPartnershipTechnique) {
00613
00614     // Retrieve the AirInv Master service context
00615     if (_airinvMasterServiceContext == NULL) {
00616         throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00617             "has not been initialised")
00618     ;
00619     }
00620     assert (_airinvMasterServiceContext != NULL);
00621
00622     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00623         *_airinvMasterServiceContext;
00624
00625     // Retrieve the slave AIRINV service object from the (AIRINV)
00626     // service context
00627     AIRINV_Service& lAIRINV_Service =
00628         lAIRINV_Master_ServiceContext.getAIRINV_Service();
00629
00630     // Retrieve the snapshot time and the airline code.
00631     const stdair::DateTime_T& lRMEventTime = iRMEvent.getRMEventTime();
00632     const stdair::AirlineCode_T& lAirlineCode = iRMEvent.getAirlineCode();
00633     const stdair::KeyDescription_T& lFDDescription =
00634         iRMEvent.getFlightDateDescription();
00635
00636     lAIRINV_Service.optimise (lAirlineCode, lFDDescription, lRMEventTime,
00637                             iForecastingMethod, iPartnershipTechnique);
00638 }

```

25.217 airinv/service/AIRINV_Master_ServiceContext.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.218 AIRINV_Master_ServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Airinv
00008 #include <airinv/basic/BasConst_AIRINV_Service.hpp>
00009 #include <airinv/service/AIRINV_Master_ServiceContext.hpp>
00010
00011 namespace AIRINV {
00012
00013     // //////////////////////////////////////
00014     AIRINV_Master_ServiceContext::AIRINV_Master_ServiceContext ()
00015         : _ownStdairService (false) {
00016     }
00017
00018     // //////////////////////////////////////

```

```

00019 AIRINV_Master_ServiceContext::~AIRINV_Master_ServiceContext() {
00020 }
00021
00022 // //////////////////////////////////////
00023 const std::string AIRINV_Master_ServiceContext::shortDisplay() const {
00024     std::ostringstream oStr;
00025     oStr << "AIRINV_Master_ServiceContext -- Owns StdAir service: "
00026         << _ownStdairService;
00027     return oStr.str();
00028 }
00029
00030 // //////////////////////////////////////
00031 const std::string AIRINV_Master_ServiceContext::display() const {
00032     std::ostringstream oStr;
00033     oStr << shortDisplay();
00034     return oStr.str();
00035 }
00036
00037 // //////////////////////////////////////
00038 const std::string AIRINV_Master_ServiceContext::describe() const {
00039     return shortDisplay();
00040 }
00041
00042 // //////////////////////////////////////
00043 void AIRINV_Master_ServiceContext::reset() {
00044     if (_ownStdairService == true) {
00045         _stdairService.reset();
00046     }
00047 }
00048
00049 }

```

25.219 airinv/service/AIRINV_Master_ServiceContext.hpp File Reference

```

#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>

```

Classes

- class [AIRINV::AIRINV_Master_ServiceContext](#)

Namespaces

- namespace [AIRINV](#)

25.220 AIRINV_Master_ServiceContext.hpp

```

00001 #ifndef __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP
00002 #define __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/service/ServiceAbstract.hpp>
00015 // AirInv
00016 #include <airinv/AIRINV_Types.hpp>
00017
00018 namespace AIRINV {
00019
00021     class AIRINV_Service;

```

```

00022 class AIRINV_Master_ServiceContext : public stdair::ServiceAbstract {
00026     friend class AIRINV_Master_Service;
00032     friend class FacAirinvMasterServiceContext;
00034
00035 private:
00036     // ////////////////////////////////// Getters //////////////////////////////////
00040     stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00041         return _stdairService;
00042     }
00043
00047     stdair::STDAIR_Service& getSTDAIR_Service() const {
00048         assert (_stdairService != NULL);
00049         return *_stdairService;
00050     }
00051
00055     const bool getOwnStdairServiceFlag() const {
00056         return _ownStdairService;
00057     }
00058
00063     AIRINV_Service& getAIRINV_Service() const {
00064         assert (_airinvService != NULL);
00065         return *_airinvService;
00066     }
00067
00068     // ////////////////////////////////// Setters //////////////////////////////////
00072     void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00073                             const bool iOwnStdairService) {
00074         _stdairService = ioSTDAIR_ServicePtr;
00075         _ownStdairService = iOwnStdairService;
00076     }
00077
00081     void setAIRINV_Service (AIRINV_ServicePtr_T ioAIRINV_ServicePtr) {
00082         _airinvService = ioAIRINV_ServicePtr;
00083     }
00084
00085
00086 private:
00087     // ////////////////////////////////// Display Methods //////////////////////////////////
00091     const std::string shortDisplay() const;
00092
00096     const std::string display() const;
00097
00101     const std::string describe() const;
00102
00103
00104 private:
00106
00109     AIRINV_Master_ServiceContext();
00113     AIRINV_Master_ServiceContext (const AIRINV_Master_ServiceContext&);
00114
00118     ~AIRINV_Master_ServiceContext();
00119
00123     void reset();
00124
00125
00126 private:
00127     // ////////////////////////////////// Children //////////////////////////////////
00131     stdair::STDAIR_ServicePtr_T _stdairService;
00132
00136     bool _ownStdairService;
00137
00138
00139 private:
00140     // ////////////////////////////////// Attributes //////////////////////////////////
00144     AIRINV_ServicePtr_T _airinvService;
00145 };
00146
00147 }
00148 #endif // __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP

```

25.221 airinv/service/AIRINV_Service.cpp File Reference

```
#include <cassert>
```

```

#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/RMOL_Service.hpp>
#include <airrac/AIRRAC_Service.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/factory/FacAirinvServiceContext.hpp>
#include <airinv/command/ScheduleParser.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>
#include <airinv/AIRINV_Service.hpp>

```

Namespaces

- namespace [AIRINV](#)

25.222 AIRINV_Service.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/make_shared.hpp>
00008 // StdAir
00009 #include <stdair/basic/BasChronometer.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BomKeyManager.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/Inventory.hpp>
00015 #include <stdair/bom/FlightDate.hpp>
00016 #include <stdair/bom/AirlineFeature.hpp>
00017 #include <stdair/bom/RMEventStruct.hpp>
00018 #include <stdair/factory/FacBomManager.hpp>
00019 #include <stdair/service/Logger.hpp>
00020 #include <stdair/STDAIR_Service.hpp>
00021 // RMOL
00022 #include <rmol/RMOL_Service.hpp>
00023 // AirRAC
00024 #include <airrac/AIRRAC_Service.hpp>
00025 // AirInv
00026 #include <airinv/basic/BasConst_AIRINV_Service.hpp>
00027 #include <airinv/factory/FacAirinvServiceContext.hpp>
00028 #include <airinv/command/ScheduleParser.hpp>
00029 #include <airinv/command/InventoryParser.hpp>
00030 #include <airinv/command/InventoryManager.hpp>
00031 #include <airinv/service/AIRINV_ServiceContext.hpp>
00032 #include <airinv/AIRINV_Service.hpp>
00033
00034 namespace AIRINV {
00035
00036 // //////////////////////////////////////
00037 AIRINV_Service::AIRINV_Service () : _airinvServiceContext (NULL) {
00038     assert (false);
00039 }
00040
00041 // //////////////////////////////////////
00042 AIRINV_Service::AIRINV_Service (const AIRINV_Service& iService)

```

```

00043 : _airinvServiceContext (NULL) {
00044     assert (false);
00045 }
00046
00047 // //////////////////////////////////////
00048 AIRINV_Service::AIRINV_Service (const stdair::BasLogParams& iLogParams)
00049 : _airinvServiceContext (NULL) {
00050
00051     // Initialise the STDAIR service handler
00052     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00053         initStdAirService (iLogParams);
00054
00055     // Initialise the service context
00056     initServiceContext();
00057
00058     // Add the StdAir service context to the AIRINV service context
00059     // \note AIRINV owns the STDAIR service resources here.
00060     const bool ownStdairService = true;
00061     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00062
00063     // Initialise the RMOL service.
00064     initRMOLService();
00065
00066     // Initialise the AIRRAC service.
00067     initAIRRACService();
00068
00069     // Initialise the (remaining of the) context
00070     initAirinvService();
00071 }
00072
00073 // //////////////////////////////////////
00074 AIRINV_Service::AIRINV_Service (const stdair::BasLogParams& iLogParams,
00075                                 const stdair::BasDBParams& iDBParams)
00076 : _airinvServiceContext (NULL) {
00077
00078     // Initialise the STDAIR service handler
00079     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00080         initStdAirService (iLogParams, iDBParams);
00081
00082     // Initialise the service context
00083     initServiceContext();
00084
00085     // Add the StdAir service context to the AIRINV service context
00086     // \note AIRINV owns the STDAIR service resources here.
00087     const bool ownStdairService = true;
00088     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00089
00090     // Initialise the RMOL service.
00091     initRMOLService();
00092
00093     // Initialise the AIRRAC service.
00094     initAIRRACService();
00095
00096     // Initialise the (remaining of the) context
00097     initAirinvService();
00098 }
00099 // //////////////////////////////////////
00100 AIRINV_Service::
00101 AIRINV_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00102 : _airinvServiceContext (NULL) {
00103
00104     // Initialise the service context
00105     initServiceContext();
00106
00107     // Store the STDAIR service object within the (AIRINV) service context
00108     // \note AirInv does not own the STDAIR service resources here.
00109     const bool doesNotOwnStdairService = false;
00110     addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00111
00112     // Initialise the RMOL service.
00113     initRMOLService();
00114
00115     // Initialise the AIRRAC service.
00116     initAIRRACService();
00117
00118     // Initialise the (remaining of the) context
00119     initAirinvService();
00120
00121 }
00122
00123 // //////////////////////////////////////
00124 AIRINV_Service::~AIRINV_Service() {
00125     // Delete/Clean all the objects from memory
00126     finalise();
00127 }
00128
00129 // //////////////////////////////////////

```

```

00130 void AIRINV_Service::finalise() {
00131     assert (_airinvServiceContext != NULL);
00132     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00133     _airinvServiceContext->reset();
00134 }
00135
00136 // //////////////////////////////////////
00137 void AIRINV_Service::initServiceContext() {
00138     // Initialise the context
00139     AIRINV_ServiceContext& lAIRINV_ServiceContext =
00140         FacAirinvServiceContext::instance().create();
00141     _airinvServiceContext = &lAIRINV_ServiceContext;
00142 }
00143
00144 // //////////////////////////////////////
00145 void AIRINV_Service::
00146 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00147                 const bool iOwnStdairService) {
00148
00149     // Retrieve the Airinv service context
00150     assert (_airinvServiceContext != NULL);
00151     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00152
00153     // Store the STDAIR service object within the (AIRINV) service context
00154     lAIRINV_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00155                                              iOwnStdairService);
00156 }
00157
00158 // //////////////////////////////////////
00159 stdair::STDAIR_ServicePtr_T AIRINV_Service::
00160 initStdAirService (const stdair::BasLogParams& iLogParams,
00161                  const stdair::BasDBParams& iDBParams) {
00162
00163     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00164         boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00165
00166     return lSTDAIR_Service_ptr;
00167 }
00168
00169 // //////////////////////////////////////
00170 stdair::STDAIR_ServicePtr_T AIRINV_Service::
00171 initStdAirService (const stdair::BasLogParams& iLogParams) {
00172
00173     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00174         boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00175
00176     return lSTDAIR_Service_ptr;
00177 }
00178
00179 // //////////////////////////////////////
00180 void AIRINV_Service::initRMOLService() {
00181
00182     // Retrieve the AirInv service context
00183     assert (_airinvServiceContext != NULL);
00184     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00185
00186     // Retrieve the StdAir service context
00187     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00188         lAIRINV_ServiceContext.getSTDAIR_ServicePtr();
00189
00190     RMOL::RMOL_ServicePtr_T lRMOL_Service_ptr =
00191         boost::make_shared<RMOL::RMOL_Service> (lSTDAIR_Service_ptr);
00192
00193     // Store the RMOL service object within the (AIRINV) service context
00194     lAIRINV_ServiceContext.setRMOL_Service (lRMOL_Service_ptr);
00195 }
00196
00197 // //////////////////////////////////////
00198 void AIRINV_Service::initAIRRACService() {
00199
00200     // Retrieve the AirInv service context
00201     assert (_airinvServiceContext != NULL);
00202     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00203
00204     // Retrieve the StdAir service context
00205     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00206         lAIRINV_ServiceContext.getSTDAIR_ServicePtr();
00207
00208     AIRRAC::AIRRAC_ServicePtr_T lAIRRAC_Service_ptr =
00209         boost::make_shared<AIRRAC::AIRRAC_Service> (lSTDAIR_Service_ptr);
00210
00211     // Store the AIRRAC service object within the (AIRINV) service context
00212     lAIRINV_ServiceContext.setAIRRAC_Service (lAIRRAC_Service_ptr);
00213 }
00214
00215 // //////////////////////////////////////
00216 void AIRINV_Service::initAirinvService() {

```

```

00245 // Do nothing at this stage. A sample BOM tree may be built by
00246 // calling the buildSampleBom() method
00247 }
00248
00249 // //////////////////////////////////////
00250 void AIRINV_Service::
00251 parseAndLoad (const stdair::Filename_T& iInventoryInputFilename) {
00252
00253 // Retrieve the BOM root object.
00254 assert (_airinvServiceContext != NULL);
00255 AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00256 stdair::STDAIR_Service& lSTDAIR_Service =
00257 lAIRINV_ServiceContext.getSTDAIR_Service();
00258 stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00259
00260 // Initialise the airline inventories
00261 InventoryParser::buildInventory (iInventoryInputFilename, lBomRoot);
00262 }
00263
00264 // //////////////////////////////////////
00265 void AIRINV_Service::
00266 parseAndLoad (const stdair::Filename_T& iScheduleInputFilename,
00267 const stdair::Filename_T& iODInputFilename,
00268 const AIRRAC::YieldFilePath& iYieldFilename) {
00269
00270 // Retrieve the BOM root object.
00271 assert (_airinvServiceContext != NULL);
00272 AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00273 stdair::STDAIR_Service& lSTDAIR_Service =
00274 lAIRINV_ServiceContext.getSTDAIR_Service();
00275 stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00276
00277 // Initialise the airline inventories
00278 ScheduleParser::generateInventories (iScheduleInputFilename, lBomRoot);
00279
00280 // Parse the yield structures.
00281 AIRRAC::AIRRAC_Service& lAIRRAC_Service =
00282 lAIRINV_ServiceContext.getAIRRAC_Service();
00283 lAIRRAC_Service.parseAndLoad (iYieldFilename);
00284
00285 // Update yield values for booking classes and O&D.
00286 lAIRRAC_Service.updateYields();
00287 }
00288
00289 // //////////////////////////////////////
00290 void AIRINV_Service::buildSampleBom() {
00291
00292 // Retrieve the AirInv service context
00293 if (_airinvServiceContext == NULL) {
00294 "
00295 "been initialised");
00296 }
00297 assert (_airinvServiceContext != NULL);
00298
00299 // Retrieve the AirInv service context and whether it owns the Stdair
00300 // service
00301 AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00302 const bool doesOwnStdairService =
00303 lAIRINV_ServiceContext.getOwnStdairServiceFlag();
00304
00305 // Retrieve the StdAir service object from the (AirInv) service context
00306 stdair::STDAIR_Service& lSTDAIR_Service =
00307 lAIRINV_ServiceContext.getSTDAIR_Service();
00308
00309 if (doesOwnStdairService == true) {
00310 //
00311 lSTDAIR_Service.buildSampleBom();
00312 }
00313
00314 AIRRAC::AIRRAC_Service& lAIRRAC_Service =
00315 lAIRINV_ServiceContext.getAIRRAC_Service();
00316 lAIRRAC_Service.buildSampleBom();
00317
00318 RMOL::RMOL_Service& lRMOL_Service= lAIRINV_ServiceContext.getRMOL_Service()
00319 ;
00320 lRMOL_Service.buildSampleBom();
00321
00322 stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00323 InventoryManager::buildSimilarSegmentCabinSets (lBomRoot);
00324
00325 // InventoryManager::setDefaultBidPriceVector (lBomRoot);
00326 }
00327
00328 // //////////////////////////////////////
00329 std::string AIRINV_Service::
00330 jsonExport (const stdair::AirlineCode_T& iAirlineCode,

```

```

00377         const stdair::FlightNumber_T& iFlightNumber,
00378         const stdair::Date_T& iDepartureDate) const {
00379
00380     // Retrieve the AIRINV service context
00381     if (_airinvServiceContext == NULL) {
00382         throw stdair::NonInitialisedServiceException ("The AirInv service "
00383             "has not been initialised")
00384     };
00385     assert (_airinvServiceContext != NULL);
00386
00387     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00388
00389     // Retrieve the STDAIR service object from the (AIRINV) service context
00390     stdair::STDAIR_Service& lSTDAIR_Service =
00391         lAIRINV_ServiceContext.getSTDAIR_Service();
00392
00393     // Delegate the JSON export to the dedicated service
00394     return lSTDAIR_Service.jsonExport (iAirlineCode, iFlightNumber,
00395         iDepartureDate);
00396 }
00397
00398 // //////////////////////////////////////
00399 std::string AIRINV_Service::
00400 list (const stdair::AirlineCode_T& iAirlineCode,
00401     const stdair::FlightNumber_T& iFlightNumber) const {
00402     std::ostringstream oFlightListStr;
00403
00404     if (_airinvServiceContext == NULL) {
00405         throw stdair::NonInitialisedServiceException ("The AirInv service "
00406             "has not been initialised")
00407     };
00408     assert (_airinvServiceContext != NULL);
00409     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00410
00411     // \todo Check that the current AIRINV_Service is actually operating for
00412     // the given airline
00413
00414     // Retrieve the STDAIR service object from the (AirInv) service context
00415     stdair::STDAIR_Service& lSTDAIR_Service =
00416         lAIRINV_ServiceContext.getSTDAIR_Service();
00417
00418     // Delegate the BOM display to the dedicated service
00419     return lSTDAIR_Service.list (iAirlineCode, iFlightNumber);
00420 }
00421
00422 // //////////////////////////////////////
00423 bool AIRINV_Service::
00424 check (const stdair::AirlineCode_T& iAirlineCode,
00425     const stdair::FlightNumber_T& iFlightNumber,
00426     const stdair::Date_T& iDepartureDate) const {
00427     std::ostringstream oFlightListStr;
00428
00429     if (_airinvServiceContext == NULL) {
00430         throw stdair::NonInitialisedServiceException ("The AirInv service "
00431             "has not been initialised")
00432     };
00433     assert (_airinvServiceContext != NULL);
00434     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00435
00436     // \todo Check that the current AIRINV_Service is actually operating for
00437     // the given airline
00438
00439     // Retrieve the STDAIR service object from the (AirInv) service context
00440     stdair::STDAIR_Service& lSTDAIR_Service =
00441         lAIRINV_ServiceContext.getSTDAIR_Service();
00442
00443     // Delegate the BOM display to the dedicated service
00444     return lSTDAIR_Service.check (iAirlineCode, iFlightNumber, iDepartureDate);
00445 }
00446
00447 // //////////////////////////////////////
00448 std::string AIRINV_Service::csvDisplay() const {
00449
00450     // Retrieve the AIRINV service context
00451     if (_airinvServiceContext == NULL) {
00452         throw stdair::NonInitialisedServiceException ("The AirInv service "
00453             "has not been initialised")
00454     };
00455     assert (_airinvServiceContext != NULL);
00456
00457     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00458
00459     // Retrieve the STDAIR service object from the (AirInv) service context

```



```

00460     stdair::STDAIR_Service& lSTDAIR_Service =
00461         lAIRINV_ServiceContext.getSTDAIR_Service();
00462
00463     // Delegate the BOM display to the dedicated service
00464     return lSTDAIR_Service.csvDisplay();
00465 }
00466
00467 // //////////////////////////////////////
00468 std::string AIRINV_Service::
00469 csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00470             const stdair::FlightNumber_T& iFlightNumber,
00471             const stdair::Date_T& iDepartureDate) const {
00472
00473     // Retrieve the AIRINV service context
00474     if (_airinvServiceContext == NULL) {
00475         throw stdair::NonInitialisedServiceException ("The AirInv service "
00476                                                         "has not been initialised")
00477     ;
00478     }
00479     assert (_airinvServiceContext != NULL);
00480     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00481
00482     // Retrieve the STDAIR service object from the (AirInv) service context
00483     stdair::STDAIR_Service& lSTDAIR_Service =
00484         lAIRINV_ServiceContext.getSTDAIR_Service();
00485
00486     // Delegate the BOM display to the dedicated service
00487     return lSTDAIR_Service.csvDisplay (iAirlineCode, iFlightNumber,
00488                                         iDepartureDate);
00489 }
00490
00491 // //////////////////////////////////////
00492 stdair::RMEventList_T AIRINV_Service::
00493 initRMEvents (const stdair::Date_T& iStartDate,
00494              const stdair::Date_T& iEndDate) {
00495
00496     if (_airinvServiceContext == NULL) {
00497         throw stdair::NonInitialisedServiceException ("The AirInv service "
00498                                                         "has not been initialised")
00499     ;
00500     }
00501     assert (_airinvServiceContext != NULL);
00502     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00503
00504     // \todo Retrieve the corresponding inventory
00505     stdair::STDAIR_Service& lSTDAIR_Service =
00506         lAIRINV_ServiceContext.getSTDAIR_Service();
00507     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00508
00509     stdair::RMEventList_T oRMEventList;
00510     const stdair::InventoryList_T& lInventoryList =
00511         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
00512     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin()
00513 ;
00514         itInv != lInventoryList.end(); ++itInv) {
00515         const stdair::Inventory* lInv_ptr = *itInv;
00516         assert (lInv_ptr != NULL);
00517
00518         InventoryManager::initRMEvents (*lInv_ptr, oRMEventList,
00519                                         iStartDate, iEndDate);
00520     }
00521     return oRMEventList;
00522 }
00523
00524 // //////////////////////////////////////
00525 void AIRINV_Service::
00526 calculateAvailability (stdair::TravelSolutionStruct& ioTravelSolution,
00527                      const stdair::PartnershipTechnique&
00528 iPartnershipTechnique) {
00529
00530     if (_airinvServiceContext == NULL) {
00531         throw stdair::NonInitialisedServiceException ("The AirInv service "
00532                                                         "has not been initialised")
00533     ;
00534     }
00535     assert (_airinvServiceContext != NULL);
00536     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00537
00538     // Retrieve the corresponding inventory.
00539     stdair::STDAIR_Service& lSTDAIR_Service =
00540         lAIRINV_ServiceContext.getSTDAIR_Service();
00541     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00542
00543     // Delegate the booking to the dedicated command
00544     stdair::BasChronometer lAvlChronometer;

```

```

00542     lAvlChronometer.start();
00543     InventoryManager::calculateAvailability (lBomRoot, ioTravelSolution,
iPartnershipTechnique);
00544     // const double lAvlMeasure = lAvlChronometer.elapsed();
00545
00546     // DEBUG
00547     // STDAIR_LOG_DEBUG ("Availability retrieval: " << lAvlMeasure << " - "
00548     //                  << lAIRINV_ServiceContext.display());
00549 }
00550
00551 // //////////////////////////////////////
00552 bool AIRINV_Service::sell (const std::string& iSegmentDateKey,
00553                          const stdair::ClassCode_T& iClassCode,
00554                          const stdair::PartySize_T& iPartySize) {
00555     bool isSellSuccessful = false;
00556
00557     if (_airinvServiceContext == NULL) {
00558         throw stdair::NonInitialisedServiceException ("The AirInv service "
00559             "has not been initialised")
;
00560     }
00561     assert (_airinvServiceContext != NULL);
00562     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00563
00564     // \todo Check that the current AIRINV_Service is actually operating for
00565     // the given airline (inventory key)
00566     // Retrieve the corresponding inventory key
00567     const stdair::InventoryKey& lInventoryKey =
00568         stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00569
00570     // Retrieve the root of the BOM tree
00571     stdair::STDAIR_Service& lSTDAIR_Service =
00572         lAIRINV_ServiceContext.getSTDAIR_Service();
00573     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00574
00575     // Retrieve the corresponding inventory
00576     stdair::Inventory& lInventory = stdair::BomManager::
00577         getObject<stdair::Inventory> (lBomRoot, lInventoryKey.toString());
00578
00579     // Delegate the booking to the dedicated command
00580     stdair::BasChronometer lSellChronometer; lSellChronometer.start();
00581     isSellSuccessful = InventoryManager::sell (lInventory, iSegmentDateKey,
00582         iClassCode, iPartySize);
00583     // const double lSellMeasure = lSellChronometer.elapsed();
00584
00585     // DEBUG
00586     // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
00587     //                  << lAIRINV_ServiceContext.display());
00588
00589     return isSellSuccessful;
00590 }
00591
00592 // //////////////////////////////////////
00593 bool AIRINV_Service::cancel (const std::string& iSegmentDateKey,
00594                             const stdair::ClassCode_T& iClassCode,
00595                             const stdair::PartySize_T& iPartySize) {
00596     bool isCancellationSuccessful = false;
00597
00598     if (_airinvServiceContext == NULL) {
00599         throw stdair::NonInitialisedServiceException ("The AirInv service "
00600             "has not been initialised")
;
00601     }
00602     assert (_airinvServiceContext != NULL);
00603     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00604
00605     // \todo Check that the current AIRINV_Service is actually operating for
00606     // the given airline (inventory key)
00607     // Retrieve the corresponding inventory key
00608     const stdair::InventoryKey& lInventoryKey =
00609         stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00610
00611     // Retrieve the root of the BOM tree
00612     stdair::STDAIR_Service& lSTDAIR_Service =
00613         lAIRINV_ServiceContext.getSTDAIR_Service();
00614     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00615
00616     // Retrieve the corresponding inventory
00617     stdair::Inventory& lInventory = stdair::BomManager::
00618         getObject<stdair::Inventory> (lBomRoot, lInventoryKey.toString());
00619
00620     // Delegate the booking to the dedicated command
00621     stdair::BasChronometer lCancellationChronometer;
00622     lCancellationChronometer.start();
00623     isCancellationSuccessful = InventoryManager::cancel (lInventory,
00624         iSegmentDateKey,
00625         iClassCode, iPartySize)

```

```

;
00626 // const double lCancellationMeasure = lCancellationChronometer.elapsed();
00627
00628 // DEBUG
00629 // STDAIR_LOG_DEBUG ("Booking cancellation: "
00630 //                  << lCancellationMeasure << " - "
00631 //                  << lAIRINV_ServiceContext.display());
00632
00633 return isCancellationSuccessful;
00634 }
00635
00636 // //////////////////////////////////////
00637 void AIRINV_Service::takeSnapshots (const stdair::AirlineCode_T& iAirlineCode
00638                                     const stdair::DateTime_T& iSnapshotTime)
00639 {
00640     if (_airinvServiceContext == NULL) {
00641         throw stdair::NonInitialisedServiceException ("The AirInv service "
00642                                                         "has not been initialised")
;
00643     }
00644     assert (_airinvServiceContext != NULL);
00645     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00646
00647     // TODO: Retrieve the corresponding inventory.
00648     stdair::STDAIR_Service& lSTDAIR_Service =
00649         lAIRINV_ServiceContext.getSTDAIR_Service();
00650     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00651
00652     const stdair::InventoryList_T lInventoryList =
00653         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
00654     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin()
;
00655         itInv != lInventoryList.end(); ++itInv) {
00656         const stdair::Inventory* lInv_ptr = *itInv;
00657         assert (lInv_ptr != NULL);
00658
00659         InventoryManager::takeSnapshots (*lInv_ptr, iSnapshotTime);
00660     }
00661 }
00662
00663 // //////////////////////////////////////
00664 void AIRINV_Service::optimise (const stdair::AirlineCode_T& iAirlineCode,
00665                               const stdair::KeyDescription_T& iFDDescription
00666                               const stdair::DateTime_T& iRMEventTime,
00667                               const stdair::ForecastingMethod&
iForecastingMethod,
00668                               const stdair::PartnershipTechnique&
iPartnershipTechnique) {
00669     if (_airinvServiceContext == NULL) {
00670         throw stdair::NonInitialisedServiceException ("The AirInv service "
00671                                                         "has not been initialised")
;
00672     }
00673     assert (_airinvServiceContext != NULL);
00674     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00675
00676     // Retrieve the corresponding inventory & flight-date
00677     stdair::STDAIR_Service& lSTDAIR_Service =
00678         lAIRINV_ServiceContext.getSTDAIR_Service();
00679     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00680     stdair::Inventory& lInventory =
00681         stdair::BomManager::getObject<stdair::Inventory> (lBomRoot, iAirlineCode)
;
00682     stdair::FlightDate& lFlightDate =
00683         stdair::BomManager::getObject<stdair::FlightDate> (lInventory,
00684                                                             iFDDescription);
00685
00686     // Retrieve the RMOL service.
00687     RMOL::RMOL_Service& lRMOL_Service = lAIRINV_ServiceContext.getRMOL_Service()
;
00688
00689     // Optimise the flight-date.
00690     bool isOptimised = lRMOL_Service.optimise (lFlightDate, iRMEventTime,
00691                                                iForecastingMethod,
iPartnershipTechnique);
00692
00693     // Update the inventory with the new controls.
00694     if (isOptimised == true) {
00695         InventoryManager::updateBookingControls (lFlightDate);
00696     }
00697 }
00698 }

```

25.223 airinv/service/AIRINV_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.224 AIRINV_ServiceContext.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // AirInv
00008 #include <airinv/basic/BasConst_AIRINV_Service.hpp>
00009 #include <airinv/service/AIRINV_ServiceContext.hpp>
00010
00011 namespace AIRINV {
00012
00013     // //////////////////////////////////////
00014     AIRINV_ServiceContext::AIRINV_ServiceContext()
00015         : _ownStdairService (false), _airlineCode (DEFAULT_AIRLINE_CODE) {
00016     }
00017
00018     // //////////////////////////////////////
00019     AIRINV_ServiceContext::
00020     AIRINV_ServiceContext (const stdair::AirlineCode_T& iAirlineCode)
00021         : _ownStdairService (false), _airlineCode (iAirlineCode) {
00022     }
00023
00024     // //////////////////////////////////////
00025     AIRINV_ServiceContext::AIRINV_ServiceContext (const AIRINV_ServiceContext&)
00026         : _ownStdairService (false), _airlineCode (DEFAULT_AIRLINE_CODE) {
00027     }
00028
00029     // //////////////////////////////////////
00030     AIRINV_ServiceContext::~AIRINV_ServiceContext() {
00031     }
00032
00033     // //////////////////////////////////////
00034     const std::string AIRINV_ServiceContext::shortDisplay() const {
00035         std::ostringstream oStr;
00036         oStr << "AIRINV_ServiceContext[" << _airlineCode
00037             << "] -- Owns StdAir service: " << _ownStdairService;
00038         return oStr.str();
00039     }
00040
00041     // //////////////////////////////////////
00042     const std::string AIRINV_ServiceContext::display() const {
00043         std::ostringstream oStr;
00044         oStr << shortDisplay();
00045         return oStr.str();
00046     }
00047
00048     // //////////////////////////////////////
00049     const std::string AIRINV_ServiceContext::describe() const {
00050         return shortDisplay();
00051     }
00052
00053     // //////////////////////////////////////
00054     void AIRINV_ServiceContext::reset() {
00055         if (_ownStdairService == true) {
00056             _stdairService.reset();
00057         }
00058     }
00059
00060 }
```

25.225 airinv/service/AIRINV_ServiceContext.hpp File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <rmol/RMOL_Types.hpp>
#include <airrac/AIRAC_Types.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class [AIRINV::AIRINV_ServiceContext](#)
Class holding the context of the AirInv services.

Namespaces

- namespace [AIRINV](#)

25.226 AIRINV_ServiceContext.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP
00002 #define __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/service/ServiceAbstract.hpp>
00014 // RMOL
00015 #include <rmol/RMOL_Types.hpp>
00016 // AIRRAC
00017 #include <airrac/AIRAC_Types.hpp>
00018 // AirInv
00019 #include <airinv/AIRINV_Types.hpp>
00020
00021 namespace AIRINV {
00022
00026 class AIRINV_ServiceContext : public stdair::ServiceAbstract {
00032     friend class AIRINV_Service;
00033     friend class FacAirinvServiceContext;
00034
00035 private:
00036     // ////////////////////////////////////// Getters //////////////////////////////////////
00040     stdair::AirlineCode_T getAirlineCode() const {
00041         return _airlineCode;
00042     }
00043
00047     stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00048         return _stdairService;
00049     }
00050
00054     stdair::STDAIR_Service& getSTDAIR_Service() const {
00055         assert (_stdairService != NULL);
00056         return *_stdairService;
00057     }
00058
00062     const bool getOwnStdairServiceFlag() const {
00063         return _ownStdairService;
00064     }
00065
00069     RMOL::RMOL_Service& getRMOL_Service() const {
00070         assert (_rmolService != NULL);
00071         return *_rmolService;
00072     }
00073
00077     AIRRAC::AIRRAC_Service& getAIRRAC_Service() const {
```

```

00078     assert (_airracService != NULL);
00079     return *_airracService;
00080 }
00081
00082
00083 private:
00084 // ////////////////////////////////// Setters //////////////////////////////////
00088 void setAirlineCode (const stdair::AirlineCode_T& iAirlineCode) {
00089     _airlineCode = iAirlineCode;
00090 }
00091
00095 void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00096     const bool iOwnStdairService) {
00097     _stdairService = ioSTDAIR_ServicePtr;
00098     _ownStdairService = iOwnStdairService;
00099 }
00100
00104 void setRMOL_Service (RMOL::RMOL_ServicePtr_T ioRMOL_ServicePtr) {
00105     _rmolService = ioRMOL_ServicePtr;
00106 }
00107
00111 void setAIRRAC_Service (AIRRAC::AIRRAC_ServicePtr_T ioAIRRAC_ServicePtr) {
00112     _airracService = ioAIRRAC_ServicePtr;
00113 }
00114
00115
00116 private:
00117 // ////////////////////////////////// Display Methods //////////////////////////////////
00121 const std::string shortDisplay() const;
00122
00126 const std::string display() const;
00127
00131 const std::string describe() const;
00132
00133
00134 private:
00136
00139 AIRINV_ServiceContext (const stdair::AirlineCode_T&);
00143 AIRINV_ServiceContext ();
00147 AIRINV_ServiceContext (const AIRINV_ServiceContext&);
00148
00152 ~AIRINV_ServiceContext ();
00153
00157 void reset();
00158
00159
00160 private:
00161 // ////////////////////////////////// Children //////////////////////////////////
00165 stdair::STDAIR_ServicePtr_T _stdairService;
00166
00170 bool _ownStdairService;
00171
00175 RMOL::RMOL_ServicePtr_T _rmolService;
00176
00180 AIRRAC::AIRRAC_ServicePtr_T _airracService;
00181
00182 private:
00183 // ////////////////////////////////// Attributes //////////////////////////////////
00188 stdair::AirlineCode_T _airlineCode;
00189 };
00190
00191 }
00192 #endif // __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP

```

25.227 airinv/service/ServiceAbstract.cpp File Reference

```
#include <airinv/service/ServiceAbstract.hpp>
```

Namespaces

- namespace [AIRINV](#)

25.228 ServiceAbstract.cpp

```

00001 // //////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // AIRINV
00005 #include <airinv/service/ServiceAbstract.hpp>
00006
00007 namespace AIRINV {
00008
00009 }

```

25.229 airinv/service/ServiceAbstract.hpp File Reference

```
#include <iosfwd>
```

Classes

- class [AIRINV::ServiceAbstract](#)

Namespaces

- namespace [AIRINV](#)

Functions

- `template<class charT , class traits >`
`std::basic_ostream< charT,`
`traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const AIRINV::ServiceAbstract &i-`
`Service)`
- `template<class charT , class traits >`
`std::basic_istream< charT,`
`traits > & operator>> (std::basic_istream< charT, traits > &ioIn, AIRINV::ServiceAbstract &ioService)`

25.229.1 Function Documentation

25.229.1.1 `template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream<`
`charT, traits > & ioOut, const AIRINV::ServiceAbstract & iService) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 42 of file [ServiceAbstract.hpp](#).

25.229.1.2 `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream<`
`charT, traits > & ioIn, AIRINV::ServiceAbstract & ioService) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 70 of file [ServiceAbstract.hpp](#).

References [AIRINV::ServiceAbstract::fromStream\(\)](#).

25.230 ServiceAbstract.hpp

```

00001 #ifndef __AIRINV_SVC_SERVICEABSTRACT_HPP
00002 #define __AIRINV_SVC_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL

```

```

00008 #include <iosfwd>
00009 // #include <sstream>
00010
00011 namespace AIRINV {
00012
00014     class ServiceAbstract {
00015     public:
00016
00018         virtual ~ServiceAbstract() {}
00019
00022         virtual void toStream (std::ostream& ioOut) const {}
00023
00026         virtual void fromStream (std::istream& ioIn) {}
00027
00028     protected:
00030         ServiceAbstract() {}
00031     };
00032 }
00033
00039 template <class charT, class traits>
00040 inline
00041 std::basic_ostream<charT, traits>&
00042 operator<< (std::basic_ostream<charT, traits>& ioOut,
00043           const AIRINV::ServiceAbstract& iService) {
00049     std::basic_ostringstream<charT, traits> ostr;
00050     ostr.copyfmt (ioOut);
00051     ostr.width (0);
00052
00053     // Fill string stream
00054     iService.toStream (ostr);
00055
00056     // Print string stream
00057     ioOut << ostr.str();
00058
00059     return ioOut;
00060 }
00061
00067 template <class charT, class traits>
00068 inline
00069 std::basic_istream<charT, traits>&
00070 operator>> (std::basic_istream<charT, traits>& ioIn,
00071            AIRINV::ServiceAbstract& ioService) {
00072     // Fill Service object with input stream
00073     ioService.fromStream (ioIn);
00074     return ioIn;
00075 }
00076
00077 #endif // __AIRINV_SVC_SERVICEABSTRACT_HPP

```

25.231 airinv/ui/cmdline/airinv.cpp File Reference

25.232 airinv.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 #include <boost/regex.hpp>
00015 #include <boost/swap.hpp>
00016 #include <boost/algorithm/string/case_conv.hpp>
00017 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/service/Logger.hpp>
00021 // AirInv
00022 #include <airinv/AIRINV_Master_Service.hpp>
00023 #include <airinv/config/airinv-paths.hpp>
00024 // GNU Readline Wrapper
00025 #include <airinv/ui/cmdline/SReadline.hpp>
00026
00027 // ////////// Constants //////////
00031 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinv.log");
00032
00036 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00037                                                         "/invdump01.csv");
00041 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00042                                                         "/schedule01.csv");

```



```

00046 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00047             "/ond01.csv");
00048
00052 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00053             "/yieldstore01.csv");
00054
00059 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00060
00065 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00066
00070 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00071
00076 typedef std::vector<std::string> TokenList_T;
00077
00081 struct Command_T {
00082     typedef enum {
00083         NOP = 0,
00084         QUIT,
00085         HELP,
00086         LIST,
00087         DISPLAY,
00088         SELECT,
00089         SELL,
00090         LAST_VALUE
00091     } Type_T;
00092 };
00093
00094 // /////////// Parsing of Options & Configuration ///////////
00095 // A helper function to simplify the main part.
00096 template<class T> std::ostream& operator<< (std::ostream& os,
00097     const std::vector<T>& v) {
00098     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00099     return os;
00100 }
00101
00105 int readConfiguration (int argc, char* argv[],
00106     bool& ioIsBuiltin, bool& ioIsForSchedule,
00107     stdair::Filename_T& ioInventoryFilename,
00108     stdair::Filename_T& ioScheduleInputFilename,
00109     stdair::Filename_T& ioODInputFilename,
00110     stdair::Filename_T& ioYieldInputFilename,
00111     std::string& ioLogFilename) {
00112     // Default for the built-in input
00113     ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00114
00115     // Default for the inventory or schedule option
00116     ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00117
00118     // Declare a group of options that will be allowed only on command line
00119     boost::program_options::options_description generic ("Generic options");
00120     generic.add_options()
00121         ("prefix", "print installation prefix")
00122         ("version,v", "print version string")
00123         ("help,h", "produce help message");
00124
00125     // Declare a group of options that will be allowed both on command
00126     // line and in config file
00127
00128     boost::program_options::options_description config ("Configuration");
00129     config.add_options()
00130         ("builtin,b",
00131             "The sample BOM tree can be either built-in or parsed from an input file.
00132             That latter must then be given with the -i/--inventory or -s/--schedule option")
00133         ("for_schedule,f",
00134             "The BOM tree should be built from a schedule file (instead of from an
00135             inventory dump)")
00136         ("inventory,i",
00137             boost::program_options::value< std::string >(&ioInventoryFilename)->
00138             default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00139             "(CSV) input file for the inventory")
00140         ("schedule,s",
00141             boost::program_options::value< std::string >(&ioScheduleInputFilename)->
00142             default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00143             "(CSV) input file for the schedule")
00144         ("ond,o",
00145             boost::program_options::value< std::string >(&ioODInputFilename)->
00146             default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00147             "(CSV) input file for the O&D")
00148         ("yield,y",
00149             boost::program_options::value< std::string >(&ioYieldInputFilename)->
00150             default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00151             "(CSV) input file for the yield")
00152         ("log,l",
00153             boost::program_options::value< std::string >(&ioLogFilename)->
00154             default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00155             "Filename for the logs")
00156     ;

```

```

00150
00151 // Hidden options, will be allowed both on command line and
00152 // in config file, but will not be shown to the user.
00153 boost::program_options::options_description hidden ("Hidden options");
00154 hidden.add_options()
00155     ("copyright",
00156      boost::program_options::value< std::vector<std::string> >(),
00157       "Show the copyright (license)");
00158
00159 boost::program_options::options_description cmdline_options;
00160 cmdline_options.add(generic).add(config).add(hidden);
00161
00162 boost::program_options::options_description config_file_options;
00163 config_file_options.add(config).add(hidden);
00164 boost::program_options::options_description visible ("Allowed options");
00165 visible.add(generic).add(config);
00166
00167 boost::program_options::positional_options_description p;
00168 p.add ("copyright", -1);
00169
00170 boost::program_options::variables_map vm;
00171 boost::program_options::
00172     store (boost::program_options::command_line_parser (argc, argv).
00173            options (cmdline_options).positional(p).run(), vm);
00174
00175 std::ifstream ifs ("airinv.cfg");
00176 boost::program_options::store (parse_config_file (ifs, config_file_options),
00177                                vm);
00178 boost::program_options::notify (vm);
00179
00180 if (vm.count ("help")) {
00181     std::cout << visible << std::endl;
00182     return K_AIRINV_EARLY_RETURN_STATUS;
00183 }
00184
00185 if (vm.count ("version")) {
00186     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00187     return K_AIRINV_EARLY_RETURN_STATUS;
00188 }
00189
00190 if (vm.count ("prefix")) {
00191     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00192     return K_AIRINV_EARLY_RETURN_STATUS;
00193 }
00194
00195 if (vm.count ("builtin")) {
00196     ioIsBuiltin = true;
00197 }
00198 const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00199 std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00200
00201 if (vm.count ("for_schedule")) {
00202     ioIsForSchedule = true;
00203 }
00204 const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00205 std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00206         << std::endl;
00207
00208 if (ioIsBuiltin == false) {
00209
00210     if (ioIsForSchedule == false) {
00211         // The BOM tree should be built from parsing an inventory dump
00212         if (vm.count ("inventory")) {
00213             ioInventoryFilename = vm["inventory"].as< std::string >();
00214             std::cout << "Input inventory filename is: " << ioInventoryFilename
00215                     << std::endl;
00216
00217         } else {
00218             // The built-in option is not selected. However, no inventory dump
00219             // file is specified
00220             std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00221                     << " -f/--for_schedule and -s/--schedule options "
00222                     << "must be specified" << std::endl;
00223         }
00224
00225     } else {
00226         // The BOM tree should be built from parsing a schedule (and O&D) file
00227         if (vm.count ("schedule")) {
00228             ioScheduleInputFilename = vm["schedule"].as< std::string >();
00229             std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00230                     << std::endl;
00231
00232         } else {
00233             // The built-in option is not selected. However, no schedule file
00234             // is specified
00235             std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00236                     << " -f/--for_schedule and -s/--schedule options "

```

```

00237         << "must be specified" << std::endl;
00238     }
00239
00240     if (vm.count ("ond")) {
00241         ioODInputFilename = vm["ond"].as< std::string >();
00242         std::cout << "Input O&D filename is: " << ioODInputFilename <<
std::endl;
00243     }
00244
00245     if (vm.count ("yield")) {
00246         ioYieldInputFilename = vm["yield"].as< std::string >();
00247         std::cout << "Input yield filename is: " << ioYieldInputFilename <<
std::endl;
00248     }
00249 }
00250 }
00251
00252 if (vm.count ("log")) {
00253     ioLogFilename = vm["log"].as< std::string >();
00254     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00255 }
00256
00257 return 0;
00258 }
00259
00260 // //////////////////////////////////////
00261 void initReadline (swift::SReadline& ioInputReader) {
00262
00263     // Prepare the list of my own completers
00264     std::vector<std::string> Completers;
00265
00266     // The following is supported:
00267     // - "identifiers"
00268     // - special identifier %file - means to perform a file name completion
00269     Completers.push_back ("help");
00270     Completers.push_back ("list %airline_code %flight_number");
00271     Completers.push_back ("select %airline_code %flight_number %flight_date");
00272     Completers.push_back ("display");
00273     Completers.push_back ("sell %booking_class %party_size %origin %destination");
00274
00275     Completers.push_back ("quit");
00276
00277     // Now register the completers.
00278     // Actually it is possible to re-register another set at any time
00279     ioInputReader.RegisterCompletions (Completers);
00280 }
00281
00282 // //////////////////////////////////////
00283 Command_T::Type_T extractCommand (TokenList_T& ioTokenList) {
00284     Command_T::Type_T oCommandType = Command_T::LAST_VALUE;
00285
00286     // Interpret the user input
00287     if (ioTokenList.empty() == false) {
00288         TokenList_T::iterator itTok = ioTokenList.begin();
00289         std::string lCommand (*itTok);
00290         boost::algorithm::to_lower (lCommand);
00291
00292         if (lCommand == "help") {
00293             oCommandType = Command_T::HELP;
00294
00295         } else if (lCommand == "list") {
00296             oCommandType = Command_T::LIST;
00297
00298         } else if (lCommand == "display") {
00299             oCommandType = Command_T::DISPLAY;
00300
00301         } else if (lCommand == "select") {
00302             oCommandType = Command_T::SELECT;
00303
00304         } else if (lCommand == "sell") {
00305             oCommandType = Command_T::SELL;
00306
00307         } else if (lCommand == "quit") {
00308             oCommandType = Command_T::QUIT;
00309         }
00310
00311         // Remove the first token (the command), as the corresponding information
00312         // has been extracted in the form of the returned command type enumeration
00313         ioTokenList.erase (itTok);
00314
00315     } else {
00316         oCommandType = Command_T::NOP;
00317     }
00318
00319     return oCommandType;
00320 }

```

```

00321
00322 ///////////////////////////////////////////////////////////////////
00323 void parseFlightKey (const TokenList_T& iTokenList,
00324                     stdair::AirlineCode_T& ioAirlineCode,
00325                     stdair::FlightNumber_T& ioFlightNumber) {
00326     // Interpret the user input
00327     if (iTokenList.empty() == false) {
00328
00329         // Read the airline code
00330         TokenList_T::const_iterator itTok = iTokenList.begin();
00331         if (itTok->empty() == false) {
00332             ioAirlineCode = *itTok;
00333             boost::algorithm::to_upper (ioAirlineCode);
00334         }
00335
00336         // Read the flight-number
00337         ++itTok;
00338         if (itTok != iTokenList.end()) {
00339
00340             if (itTok->empty() == false) {
00341                 try {
00342
00343                     ioFlightNumber = boost::lexical_cast<stdair::FlightNumber_T> (*itTok)
;
00344
00345                 } catch (boost::bad_lexical_cast& eCast) {
00346                     std::cerr << "The flight number ('" << *itTok
00347                         << "') cannot be understood. "
00348                         << "The default value (all) is kept."
00349                         << std::endl;
00350                     return;
00351                 }
00352             }
00353
00354         } else {
00355             return;
00356         }
00357     }
00358 }
00359
00360 ///////////////////////////////////////////////////////////////////
00361 void parseFlightDateKey (const TokenList_T& iTokenList,
00362                          stdair::AirlineCode_T& ioAirlineCode,
00363                          stdair::FlightNumber_T& ioFlightNumber,
00364                          stdair::Date_T& ioDepartureDate) {
00365     //
00366     const std::string kMonthStr[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
00367                                         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
00368     //
00369     unsigned short ioDepartureDateYear = ioDepartureDate.year();
00370     unsigned short ioDepartureDateMonth = ioDepartureDate.month();
00371     std::string ioDepartureDateMonthStr = kMonthStr[ioDepartureDateMonth-1];
00372     unsigned short ioDepartureDateDay = ioDepartureDate.day();
00373
00374     // Interpret the user input
00375     if (iTokenList.empty() == false) {
00376
00377         // Read the airline code
00378         TokenList_T::const_iterator itTok = iTokenList.begin();
00379         if (itTok->empty() == false) {
00380             ioAirlineCode = *itTok;
00381             boost::algorithm::to_upper (ioAirlineCode);
00382         }
00383
00384         // Read the flight-number
00385         ++itTok;
00386         if (itTok != iTokenList.end()) {
00387
00388             if (itTok->empty() == false) {
00389                 try {
00390
00391                     ioFlightNumber = boost::lexical_cast<stdair::FlightNumber_T> (*itTok)
;
00392
00393                 } catch (boost::bad_lexical_cast& eCast) {
00394                     std::cerr << "The flight number ('" << *itTok
00395                         << "') cannot be understood. "
00396                         << "The default value (all) is kept."
00397                         << std::endl;
00398                     return;
00399                 }
00400             }
00401
00402         } else {
00403             return;
00404         }
00405

```

```

00406 // Read the year for the departure date
00407 ++itTok;
00408 if (itTok != iTokenList.end()) {
00409     if (itTok->empty() == false) {
00410         try {
00411             ioDepartureDateYear = boost::lexical_cast<unsigned short> (*itTok);
00412             if (ioDepartureDateYear < 100) {
00413                 ioDepartureDateYear += 2000;
00414             }
00415         } catch (boost::bad_lexical_cast& eCast) {
00416             std::cerr << "The year of the flight departure date ('" << *itTok
00417                 << "') cannot be understood. The default value ("
00418                 << ioDepartureDateYear << ") is kept. " << std::endl;
00419             return;
00420         }
00421     } else {
00422         return;
00423     }
00424 }
00425 // Read the month for the departure date
00426 ++itTok;
00427 if (itTok != iTokenList.end()) {
00428     if (itTok->empty() == false) {
00429         try {
00430             const boost::regex lMonthRegex ("^(\\d{1,2})$");
00431             const bool isMonthANumber = regex_match (*itTok, lMonthRegex);
00432             if (isMonthANumber == true) {
00433                 const unsigned short lMonth =
00434                     boost::lexical_cast<unsigned short> (*itTok);
00435                 if (lMonth > 12) {
00436                     throw boost::bad_lexical_cast();
00437                 }
00438                 ioDepartureDateMonthStr = kMonthStr[lMonth-1];
00439             } else {
00440                 const std::string lMonthStr (*itTok);
00441                 if (lMonthStr.size() < 3) {
00442                     throw boost::bad_lexical_cast();
00443                 }
00444                 std::string lMonthStr1 (lMonthStr.substr (0, 1));
00445                 boost::algorithm::to_upper (lMonthStr1);
00446                 std::string lMonthStr23 (lMonthStr.substr (1, 2));
00447                 boost::algorithm::to_lower (lMonthStr23);
00448                 ioDepartureDateMonthStr = lMonthStr1 + lMonthStr23;
00449             }
00450         } catch (boost::bad_lexical_cast& eCast) {
00451             std::cerr << "The month of the flight departure date ('" << *itTok
00452                 << "') cannot be understood. The default value ("
00453                 << ioDepartureDateMonthStr << ") is kept. " << std::endl;
00454             return;
00455         }
00456     } else {
00457         return;
00458     }
00459 }
00460 // Read the day for the departure date
00461 ++itTok;
00462 if (itTok != iTokenList.end()) {
00463     if (itTok->empty() == false) {
00464         try {
00465             ioDepartureDateDay = boost::lexical_cast<unsigned short> (*itTok);
00466         } catch (boost::bad_lexical_cast& eCast) {
00467             std::cerr << "The day of the flight departure date ('" << *itTok
00468                 << "') cannot be understood. The default value ("
00469                 << ioDepartureDateDay << ") is kept. " << std::endl;
00470             return;
00471         }
00472     } else {
00473         return;
00474     }
00475 }
00476 }
00477 }
00478 }
00479 }
00480 }
00481 }
00482 }
00483 }
00484 }
00485 }
00486 }
00487 }
00488 }
00489 }
00490 }
00491 }
00492 }

```

```

00493 // Re-compose the departure date
00494 std::ostringstream lDepartureDateStr;
00495 lDepartureDateStr << ioDepartureDateYear << "-" << ioDepartureDateMonthStr
00496 << "-" << ioDepartureDateDay;
00497
00498 try {
00499     ioDepartureDate =
00500         boost::gregorian::from_simple_string (lDepartureDateStr.str());
00501
00502 } catch (boost::gregorian::bad_month& eCast) {
00503     std::cerr << "The flight departure date ('" << lDepartureDateStr.str()
00504 << "') cannot be understood. The default value ("
00505 << ioDepartureDate << ") is kept. " << std::endl;
00506     return;
00507 }
00508 }
00509 }
00510 }
00511 }
00512
00513 // //////////////////////////////////////
00514 void parseBookingClassKey (const TokenList_T& iTokenList,
00515                             stdair::ClassCode_T& ioBookingClass,
00516                             stdair::PartySize_T& ioPartySize,
00517                             stdair::AirportCode_T& ioOrigin,
00518                             stdair::AirportCode_T& ioDestination) {
00519     // Interpret the user input
00520     if (iTokenList.empty() == false) {
00521
00522         // Read the booking class
00523         TokenList_T::const_iterator itTok = iTokenList.begin();
00524         if (itTok->empty() == false) {
00525             ioBookingClass = *itTok;
00526             boost::algorithm::to_upper (ioBookingClass);
00527         }
00528
00529         // Read the party size
00530         ++itTok;
00531         if (itTok != iTokenList.end()) {
00532
00533             if (itTok->empty() == false) {
00534                 try {
00535
00536                     ioPartySize = boost::lexical_cast<stdair::PartySize_T> (*itTok);
00537
00538                 } catch (boost::bad_lexical_cast& eCast) {
00539                     std::cerr << "The party size ('" << *itTok
00540 << "') cannot be understood. The default value ("
00541 << ioPartySize << ") is kept." << std::endl;
00542                     return;
00543                 }
00544             }
00545
00546         } else {
00547             return;
00548         }
00549
00550         // Read the origin
00551         ++itTok;
00552         if (itTok != iTokenList.end()) {
00553
00554             if (itTok->empty() == false) {
00555                 ioOrigin = *itTok;
00556                 boost::algorithm::to_upper (ioOrigin);
00557             }
00558
00559         } else {
00560             return;
00561         }
00562
00563         // Read the destination
00564         ++itTok;
00565         if (itTok != iTokenList.end()) {
00566
00567             if (itTok->empty() == false) {
00568                 ioDestination = *itTok;
00569                 boost::algorithm::to_upper (ioDestination);
00570             }
00571
00572         } else {
00573             return;
00574         }
00575     }
00576 }
00577
00578 // //////////////////////////////////////
00579 std::string toString (const TokenList_T& iTokenList) {

```

```

00580     std::ostringstream oStr;
00581
00582     // Re-create the string with all the tokens, trimmed by read-line
00583     unsigned short idx = 0;
00584     for (TokenList_T::const_iterator iTok = iTokenList.begin();
00585          iTok != iTokenList.end(); ++iTok, ++idx) {
00586         if (idx != 0) {
00587             oStr << " ";
00588         }
00589         oStr << *iTok;
00590     }
00591
00592     return oStr.str();
00593 }
00594
00595 // //////////////////////////////////////
00596 TokenList_T extractTokenList (const TokenList_T& iTokenList,
00597                               const std::string& iRegularExpression) {
00598     TokenList_T oTokenList;
00599
00600     // Re-create the string with all the tokens (which had been trimmed
00601     // by read-line)
00602     const std::string lFullLine = toString (iTokenList);
00603
00604     // See the caller for the regular expression
00605     boost::regex expression (iRegularExpression);
00606
00607     std::string::const_iterator start = lFullLine.begin();
00608     std::string::const_iterator end = lFullLine.end();
00609
00610     boost::match_results<std::string::const_iterator> what;
00611     boost::match_flag_type flags = boost::match_default | boost::format_sed;
00612     regex_search (start, end, what, expression, flags);
00613
00614     // Put the matched strings in the list of tokens to be returned back
00615     // to the caller
00616     const unsigned short lMatchSetSize = what.size();
00617     for (unsigned short matchIdx = 1; matchIdx != lMatchSetSize; ++matchIdx) {
00618         const std::string lMatchedString (std::string (what[matchIdx].first,
00619                                                         what[matchIdx].second));
00620         //if (lMatchedString.empty() == false) {
00621             oTokenList.push_back (lMatchedString);
00622         //}
00623     }
00624
00625     // DEBUG
00626     // std::cout << "After (token list): " << oTokenList << std::endl;
00627
00628     return oTokenList;
00629 }
00630
00631 // //////////////////////////////////////
00632 TokenList_T extractTokenListForFlight (const TokenList_T& iTokenList) {
00633     const std::string lRegex ("^([[:alpha:]]{2,3})?"
00634                               "[[:space:]]*([[:digit:]]{1,4})?$");
00635
00636     //
00637     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00638     return oTokenList;
00639 }
00640
00641 // //////////////////////////////////////
00642 TokenList_T extractTokenListForFlightDate (const TokenList_T& iTokenList) {
00643     const std::string lRegex ("^([[:alpha:]]{2,3})?"
00644                               "[[:space:]]*([[:digit:]]{1,4})?"
00645                               "[/ ]*"
00646                               "([[:digit:]]{2,4})?[/-]?[[:space:]]*"
00647                               "([[:alpha:]]{3}|[[:digit:]]{1,2})?[/-]?[[:space:]]*"
00648                               "([[:digit:]]{1,2})?$");
00649
00650     //
00651     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00652     return oTokenList;
00653 }
00654
00655 // //////////////////////////////////////
00656 TokenList_T extractTokenListForClass (const TokenList_T& iTokenList) {
00657     const std::string lRegex ("^([[:alpha:]]?"
00658                               "[[:space:]]*([[:digit:]]{1,3})?"
00659                               "[[:space:]]*([[:alpha:]]{3})?"
00660                               "[[:space:]]*([[:alpha:]]{3})?$");
00661
00662     //
00663     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00664     return oTokenList;
00665 }
00666
00667 //
00668 const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00669 return oTokenList;
00670 }
00671
00672 // //////////////////////////////////////
00673 TokenList_T extractTokenListForClass (const TokenList_T& iTokenList) {
00674     const std::string lRegex ("^([[:alpha:]]?"
00675                               "[[:space:]]*([[:digit:]]{1,3})?"
00676                               "[[:space:]]*([[:alpha:]]{3})?"
00677                               "[[:space:]]*([[:alpha:]]{3})?$");
00678
00679     //
00680     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00681     return oTokenList;
00682 }
00683
00684 //
00685 const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00686 return oTokenList;
00687 }
00688
00689 //
00690 const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00691 return oTokenList;
00692 }

```

```

00690
00691
00692 // //////////// M A I N ////////////
00693 int main (int argc, char* argv[]) {
00694
00695     // State whether the BOM tree should be built-in or parsed from an
00696     // input file
00697     bool isBuiltin;
00698     bool isForSchedule;
00699
00700     // Input file names
00701     stdair::Filename_T lInventoryFilename;
00702     stdair::Filename_T lScheduleInputFilename;
00703     stdair::Filename_T lODInputFilename;
00704     stdair::Filename_T lYieldInputFilename;
00705
00706     // Readline history
00707     const unsigned int lHistorySize (100);
00708     const std::string lHistoryFilename ("airinv.hist");
00709     const std::string lHistoryBackupFilename ("airinv.hist.bak");
00710
00711     // Default parameters for the interactive session
00712     stdair::AirlineCode_T lLastInteractiveAirlineCode;
00713     stdair::FlightNumber_T lLastInteractiveFlightNumber;
00714     stdair::Date_T lLastInteractiveDate;
00715     stdair::AirlineCode_T lInteractiveAirlineCode;
00716     stdair::FlightNumber_T lInteractiveFlightNumber;
00717     stdair::Date_T lInteractiveDate;
00718     stdair::AirportCode_T lInteractiveOrigin;
00719     stdair::AirportCode_T lInteractiveDestination;
00720     stdair::ClassCode_T lInteractiveBookingClass;
00721     stdair::PartySize_T lInteractivePartySize;
00722
00723     // Parameters for the sale
00724     std::string lSegmentDateKey;
00725
00726     // Output log File
00727     stdair::Filename_T lLogFilename;
00728
00729     // Call the command-line option parser
00730     const int lOptionParserStatus =
00731         readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
00732
00733             , lScheduleInputFilename, lODInputFilename,
00734             lYieldInputFilename, lLogFilename);
00735
00736     if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00737         return 0;
00738     }
00739
00740     // Set the log parameters
00741     std::ofstream logOutputFile;
00742     // Open and clean the log outputfile
00743     logOutputFile.open (lLogFilename.c_str());
00744     logOutputFile.clear();
00745
00746     // Initialise the inventory service
00747     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00748     AIRINV::AIRINV_Master_Service airinvService (lLogParams);
00749
00750     // DEBUG
00751     STDAIR_LOG_DEBUG ("Welcome to AirInv");
00752
00753     // Check whether or not a (CSV) input file should be read
00754     if (isBuiltin == true) {
00755         // Build the sample BOM tree for RMOL
00756         airinvService.buildSampleBom();
00757
00758         // Update the default parameters for the following interactive session
00759         lInteractiveAirlineCode = "BA";
00760         lInteractiveFlightNumber = 9;
00761         lInteractiveDate = stdair::Date_T (2011, 06, 10);
00762         lInteractiveBookingClass = "Q";
00763         lInteractivePartySize = 2;
00764         lInteractiveOrigin = "LHR";
00765         lInteractiveDestination = "SYD";
00766     } else {
00767         if (isForSchedule == true) {
00768             // Build the BOM tree from parsing a schedule file (and O&D list)
00769             AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00770             airinvService.parseAndLoad (lScheduleInputFilename, lODInputFilename,
00771                                     lYieldFilePath);
00772
00773             // Update the default parameters for the following interactive session
00774             lInteractiveAirlineCode = "SQ";
00775

```



```

00776     lInteractiveFlightNumber = 11;
00777     lInteractiveDate = stdair::Date_T (2010, 01, 15);
00778     lInteractiveBookingClass = "Y";
00779     lInteractivePartySize = 2;
00780     lInteractiveOrigin = "SIN";
00781     lInteractiveDestination = "BKK";
00782
00783 } else {
00784     // Build the BOM tree from parsing an inventory dump file
00785     airinvService.parseAndLoad (lInventoryFilename);
00786
00787     // Update the default parameters for the following interactive session
00788     lInteractiveAirlineCode = "SV";
00789     lInteractiveFlightNumber = 5;
00790     lInteractiveDate = stdair::Date_T (2010, 03, 11);
00791     lInteractiveBookingClass = "Y";
00792     lInteractivePartySize = 2;
00793     lInteractiveOrigin = "KBP";
00794     lInteractiveDestination = "JFK";
00795 }
00796 }
00797
00798 // Save the last state
00799 lLastInteractiveAirlineCode = lInteractiveAirlineCode;
00800 lLastInteractiveFlightNumber = lInteractiveFlightNumber;
00801 lLastInteractiveDate = lInteractiveDate;
00802
00803 // DEBUG
00804 STDAIR_LOG_DEBUG ("=====");
00805 STDAIR_LOG_DEBUG ("=          Beginning of the interactive session          =");
00806 STDAIR_LOG_DEBUG ("=====");
00807
00808 // Initialise the GNU readline wrapper
00809 swift::SReadline lReader (lHistoryFilename, lHistorySize);
00810 initReadline (lReader);
00811
00812 // Now we can ask user for a line
00813 std::string lUserInput;
00814 bool EndOfInput (false);
00815 Command_T::Type_T lCommandType (Command_T::NOP);
00816
00817 while (lCommandType != Command_T::QUIT && EndOfInput == false) {
00818     // Prompt
00819     std::ostream oPromptStr;
00820     oPromptStr << "airinv "
00821     << lInteractiveAirlineCode << lInteractiveFlightNumber
00822     << " / " << lInteractiveDate
00823     << "> ";
00824     // Call read-line, which will fill the list of tokens
00825     TokenList_T lTokenListByReadline;
00826     lUserInput = lReader.GetLine (oPromptStr.str(), lTokenListByReadline,
00827     EndOfInput);
00828
00829     // The history can be saved to an arbitrary file at any time
00830     lReader.SaveHistory (lHistoryBackupFilename);
00831
00832     // The end-of-input typically corresponds to a CTRL-D typed by the user
00833     if (EndOfInput) {
00834         std::cout << std::endl;
00835         break;
00836     }
00837
00838     // Interpret the user input
00839     lCommandType = extractCommand (lTokenListByReadline);
00840
00841     switch (lCommandType) {
00842
00843         // ////////////////////////////////// Help //////////////////////////////////
00844         case Command_T::HELP: {
00845             std::cout << std::endl;
00846             std::cout << "Commands: " << std::endl;
00847             std::cout << " help" << "\t\t" << "Display this help" << std::endl;
00848             std::cout << " quit" << "\t\t" << "Quit the application" << std::endl;
00849             std::cout << " list" << "\t\t"
00850             << "List airlines, flights and departure dates" << std::endl;
00851             std::cout << " select" << "\t\t"
00852             << "Select a flight-date to become the current one"
00853             << std::endl;
00854             std::cout << " display" << "\t\t"
00855             << "Display the current flight-date" << std::endl;
00856             std::cout << " sell" << "\t\t"
00857             << "Make a booking on the current flight-date" << std::endl;
00858             std::cout << std::endl;
00859             break;
00860         }
00861
00862         // ////////////////////////////////// Quit //////////////////////////////////

```

```

00863     case Command_T::QUIT: {
00864         break;
00865     }
00866
00867     // ////////////////////////////////// List //////////////////////////////////
00868     case Command_T::LIST: {
00869         //
00870         TokenList_T lTokenList = extractTokenListForFlight (lTokenListByReadline)
00871 ;
00872         stdair::AirlineCode_T lAirlineCode ("all");
00873         stdair::FlightNumber_T lFlightNumber (0);
00874         // Parse the parameters given by the user, giving default values
00875         // in case the user does not specify some (or all) of them
00876         parseFlightKey (lTokenList, lAirlineCode, lFlightNumber);
00877
00878         //
00879         const std::string lFlightNumberStr = (lFlightNumber ==0)?" (all)": "";
00880         std::cout << "List of flights for "
00881                 << lAirlineCode << " " << lFlightNumber << lFlightNumberStr
00882                 << std::endl;
00883
00884         // DEBUG: Display the flight-date
00885         const std::string& lFlightDateListStr =
00886             airinvService.list (lAirlineCode, lFlightNumber);
00887
00888         if (lFlightDateListStr.empty() == false) {
00889             std::cout << lFlightDateListStr << std::endl;
00890             STDAIR_LOG_DEBUG (lFlightDateListStr);
00891
00892         } else {
00893             std::cerr << "There is no result for "
00894                     << lAirlineCode << " " << lFlightNumber << lFlightNumberStr
00895                     << ". Just type the list command without any parameter "
00896                     << "to see the flight-dates for all the airlines and for all
00897
00898                     << "the flight numbers."
00899                     << std::endl;
00900         }
00901         break;
00902     }
00903
00904     // ////////////////////////////////// Select //////////////////////////////////
00905     case Command_T::SELECT: {
00906         //
00907         TokenList_T lTokenList =
00908             extractTokenListForFlightDate (lTokenListByReadline);
00909
00910         // Check whether the user wants to select the last saved flight-date
00911         if (lTokenList.empty() == false) {
00912             // Read the booking class
00913             TokenList_T::const_iterator itTok = lTokenList.begin();
00914
00915             if (*itTok == "-") {
00916
00917                 // Swap the current state with the last state
00918                 boost::swap (lInteractiveAirlineCode, lLastInteractiveAirlineCode);
00919                 boost::swap (lInteractiveFlightNumber, lLastInteractiveFlightNumber);
00920                 boost::swap (lInteractiveDate, lLastInteractiveDate);
00921
00922                 break;
00923             }
00924         }
00925
00926         // Parse the parameters given by the user, giving default values
00927         // in case the user does not specify some (or all) of them
00928         parseFlightDateKey (lTokenList, lInteractiveAirlineCode,
00929                             lInteractiveFlightNumber, lInteractiveDate);
00930
00931         // Check whether the selected flight-date is valid
00932         const bool isFlightDateValid =
00933             airinvService.check (lInteractiveAirlineCode, lInteractiveFlightNumber,
00934                                 lInteractiveDate);
00935         if (isFlightDateValid == false) {
00936             std::ostringstream oFDKStr;
00937             oFDKStr << "The " << lInteractiveAirlineCode
00938                     << lInteractiveFlightNumber << " / " << lInteractiveDate
00939                     << " flight-date is not valid. Make sure it exists (e.g., "
00940                     << " with the list command). The current flight-date is kept "
00941                     << " selected.";
00942             std::cout << oFDKStr.str() << std::endl;
00943             STDAIR_LOG_ERROR (oFDKStr.str());
00944
00945             // Restore the last state
00946             lInteractiveAirlineCode = lLastInteractiveAirlineCode;
00947             lInteractiveFlightNumber = lLastInteractiveFlightNumber;

```

```

00948         lInteractiveDate = lLastInteractiveDate;
00949
00950         break;
00951     }
00952
00953     // DEBUG: Display the flight-date selection
00954     std::ostringstream oFDKStr;
00955     oFDKStr << "Selected the " << lInteractiveAirlineCode
00956         << lInteractiveFlightNumber << " / " << lInteractiveDate
00957         << " flight-date";
00958     std::cout << oFDKStr.str() << std::endl;
00959     STDAIR_LOG_DEBUG (oFDKStr.str());
00960
00961     // Save the last state
00962     lLastInteractiveAirlineCode = lInteractiveAirlineCode;
00963     lLastInteractiveFlightNumber = lInteractiveFlightNumber;
00964     lLastInteractiveDate = lInteractiveDate;
00965
00966     break;
00967 }
00968
00969 // ////////////////////////////////// Display //////////////////////////////////
00970 case Command_T::DISPLAY: {
00971     // DEBUG: Display the flight-date
00972     const std::string& lCSVFlightDateDump =
00973         airinvService.csvDisplay (lInteractiveAirlineCode,
00974             lInteractiveFlightNumber, lInteractiveDate);
00975     std::cout << lCSVFlightDateDump << std::endl;
00976     STDAIR_LOG_DEBUG (lCSVFlightDateDump);
00977
00978     break;
00979 }
00980
00981 // ////////////////////////////////// Sell //////////////////////////////////
00982 case Command_T::SELL: {
00983     //
00984     TokenList_T lTokenList = extractTokenListForClass (lTokenListByReadline);
00985
00986     // Parse the parameters given by the user, giving default values
00987     // in case the user does not specify some (or all) of them
00988     parseBookingClassKey (lTokenList, lInteractiveBookingClass,
00989         lInteractivePartySize,
00990         lInteractiveOrigin, lInteractiveDestination);
00991
00992     // DEBUG: Display the flight-date before the sell
00993     const std::string& lCSVFlightDateDumpBefore =
00994         airinvService.csvDisplay (lInteractiveAirlineCode,
00995             lInteractiveFlightNumber, lInteractiveDate);
00996     //std::cout << lCSVFlightDateDumpBefore << std::endl;
00997     STDAIR_LOG_DEBUG (lCSVFlightDateDumpBefore);
00998
00999     // Make a booking
01000     std::ostringstream oSDKStr;
01001     oSDKStr << lInteractiveAirlineCode << ","
01002         << lInteractiveFlightNumber << ","
01003         << lInteractiveDate << ","
01004         << lInteractiveOrigin << "," << lInteractiveDestination;
01005     const std::string lSegmentDateKey (oSDKStr.str());
01006
01007     // Perform the sell
01008     const bool isSellSuccessful =
01009         airinvService.sell (lSegmentDateKey,
01010             lInteractiveBookingClass, lInteractivePartySize);
01011
01012     // DEBUG
01013     const std::string isSellSuccessfulStr =
01014         (isSellSuccessful == true)?"Yes":"No";
01015     std::ostringstream oSaleStr;
01016     oSaleStr << "Sale (" << lSegmentDateKey << ", "
01017         << lInteractiveBookingClass << ": " << lInteractivePartySize
01018         << ") successful? " << isSellSuccessfulStr;
01019     std::cout << oSaleStr.str() << std::endl;
01020
01021     // DEBUG
01022     STDAIR_LOG_DEBUG (oSaleStr.str());
01023
01024     // DEBUG: Display the flight-date after the sell
01025     const std::string& lCSVFlightDateDumpAfter =
01026         airinvService.csvDisplay (lInteractiveAirlineCode,
01027             lInteractiveFlightNumber, lInteractiveDate);
01028     //std::cout << lCSVFlightDateDumpAfter << std::endl;
01029     STDAIR_LOG_DEBUG (lCSVFlightDateDumpAfter);
01030
01031     break;
01032 }
01033
01034 // ////////////////////////////////// Default / No value //////////////////////////////////

```

```

01035     case Command_T::NOP: {
01036         break;
01037     }
01038
01039     case Command_T::LAST_VALUE:
01040     default: {
01041         // DEBUG
01042         std::ostringstream ostr;
01043         ostr << "That command is not yet understood: '" << lUserInput
01044             << "' => " << lTokenListByReadline;
01045         STDAIR_LOG_DEBUG (ostr.str());
01046         std::cout << ostr.str() << std::endl;
01047     }
01048 }
01049 }
01050
01051 // DEBUG
01052 STDAIR_LOG_DEBUG ("End of the session. Exiting.");
01053 std::cout << "End of the session. Exiting." << std::endl;
01054
01055 // Close the Log outputFile
01056 logOutputFile.close();
01057
01058 /*
01059 Note: as that program is not intended to be run on a server in
01060 production, it is better not to catch the exceptions. When it
01061 happens (that an exception is throwned), that way we get the
01062 call stack.
01063 */
01064
01065 return 0;
01066 }

```

25.233 airinv/ui/cmdline/readline_autocomp.hpp File Reference

```

#include <string>
#include <iosfwd>
#include <cstdio>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <readline/readline.h>
#include <readline/history.h>

```

Classes

- struct [COMMAND](#)

Typedefs

- typedef int(* [pt2Func](#))(char *)

Functions

- char * [getwd](#) ()
- char * [xmalloc](#) (size_t)
- int [com_list](#) (char *)
- int [com_view](#) (char *)
- int [com_rename](#) (char *)
- int [com_stat](#) (char *)
- int [com_pwd](#) (char *)
- int [com_delete](#) (char *)
- int [com_help](#) (char *)

- int [com_cd](#) (char *)
- int [com_quit](#) (char *)
- char * [stripwhite](#) (char *iString)
- [COMMAND](#) * [find_command](#) (char *iString)
- char * [dupstr](#) (char *iString)
- int [execute_line](#) (char *line)
- char * [command_generator](#) (char *text, int state)
- char ** [fileman_completion](#) (char *text, int start, int end)
- void [initialize_readline](#) ()
- void [too_dangerous](#) (char *caller)
- int [valid_argument](#) (char *caller, char *arg)

Variables

- [COMMAND](#) [commands](#) []
- int [done](#)
- static char [syscom](#) [1024]

25.233.1 Typedef Documentation

25.233.1.1 typedef int(* [pt2Func](#))(char *)

Definition at line 35 of file [readline_autocomp.hpp](#).

25.233.2 Function Documentation

25.233.2.1 char* [getwd](#) ()

[readline_autocomp.hpp](#) -- A tiny application which demonstrates how to use the GNU Readline library. This application interactively allows users to manipulate files and their modes.

Referenced by [com_pwd\(\)](#).

25.233.2.2 char* [xmalloc](#) (size_t)

Referenced by [dupstr\(\)](#).

25.233.2.3 void [com_list](#) (char * *arg*)

List the file(s) named in arg.

Definition at line 264 of file [readline_autocomp.hpp](#).

25.233.2.4 int [com_view](#) (char * *arg*)

Definition at line 274 of file [readline_autocomp.hpp](#).

References [valid_argument\(\)](#).

25.233.2.5 int [com_rename](#) (char * *arg*)

Definition at line 284 of file [readline_autocomp.hpp](#).

References [too_dangerous\(\)](#).

25.233.2.6 int [com_stat](#) (char * *arg*)

Definition at line 289 of file [readline_autocomp.hpp](#).

References [valid_argument\(\)](#).

25.233.2.7 int com_pwd (char * *ignore*)

Definition at line 367 of file [readline_autocomp.hpp](#).

References [getwd\(\)](#).

Referenced by [com_cd\(\)](#).

25.233.2.8 int com_delete (char * *arg*)

Definition at line 315 of file [readline_autocomp.hpp](#).

References [too_dangerous\(\)](#).

25.233.2.9 int com_help (char * *arg*)

Print out help for ARG, or for all of the commands if ARG is not present.

Definition at line 324 of file [readline_autocomp.hpp](#).

References [COMMAND::name](#).

25.233.2.10 int com_cd (char * *arg*)

Definition at line 356 of file [readline_autocomp.hpp](#).

References [com_pwd\(\)](#).

25.233.2.11 int com_quit (char * *arg*)

Definition at line 381 of file [readline_autocomp.hpp](#).

25.233.2.12 char * stripwhite (char * *string*)

Strip whitespace from the start and end of STRING. Return a pointer into STRING.

Definition at line 152 of file [readline_autocomp.hpp](#).

25.233.2.13 COMMAND * find_command (char * *name*)

Look up NAME as the name of a command, and return a pointer to that command. Return a NULL pointer if NAME isn't a command name.

Definition at line 136 of file [readline_autocomp.hpp](#).

References [COMMAND::name](#).

Referenced by [execute_line\(\)](#).

25.233.2.14 char* dupstr (char * *iString*)

Duplicate a string

Definition at line 85 of file [readline_autocomp.hpp](#).

References [xmalloc\(\)](#).

Referenced by [command_generator\(\)](#).

25.233.2.15 int execute_line (char * *line*)

Execute a command line.

Definition at line 94 of file [readline_autocomp.hpp](#).

References [find_command\(\)](#), and [COMMAND::func](#).

25.233.2.16 `char * command_generator (char * text, int state)`

Generator function for command completion. STATE lets us know whether to start from scratch; without any state (i.e. STATE == 0), then we start at the top of the list.

Definition at line 222 of file [readline_autocomp.hpp](#).

References [dupstr\(\)](#).

Referenced by [fileman_completion\(\)](#).

25.233.2.17 `char ** fileman_completion (char * text, int start, int end)`

Attempt to complete on the contents of TEXT. START and END bound the region of rl_line_buffer that contains the word to complete. TEXT is the word to complete. We can use the entire contents of rl_line_buffer in case we want to do some simple parsing. Return the array of matches, or NULL if there aren't any.

Definition at line 200 of file [readline_autocomp.hpp](#).

References [command_generator\(\)](#).

Referenced by [initialize_readline\(\)](#).

25.233.2.18 `void initialize_readline ()`

Tell the GNU Readline library how to complete. We want to try to complete on command names if this is the first word in the line, or on filenames if not.

Definition at line 185 of file [readline_autocomp.hpp](#).

References [fileman_completion\(\)](#).

25.233.2.19 `void too_dangerous (char * caller)`

Definition at line 387 of file [readline_autocomp.hpp](#).

Referenced by [com_delete\(\)](#), and [com_rename\(\)](#).

25.233.2.20 `int valid_argument (char * caller, char * arg)`

Definition at line 395 of file [readline_autocomp.hpp](#).

Referenced by [com_stat\(\)](#), and [com_view\(\)](#).

25.233.3 Variable Documentation**25.233.3.1** `COMMAND commands[]`

Initial value:

```
{
  { "cd", (*com_cd)(), "Change to directory DIR" },
  { "delete", com_delete, "Delete FILE" },
  { "help", com_help, "Display this text" },
  { "?", com_help, "Synonym for 'help'" },
  { "list", com_list, "List files in DIR" },
  { "ls", com_list, "Synonym for 'list'" },
  { "pwd", com_pwd, "Print the current working directory" },
  { "quit", com_quit, "Quit using airinv" },
  { "rename", com_rename, "Rename FILE to NEWNAME" },
  { "stat", com_stat, "Print out statistics on FILE" },
  { "view", com_view, "View the contents of FILE" },
  { (char*) NULL, (pt2Func) NULL, (char*) NULL }
}
```

Definition at line 58 of file [readline_autocomp.hpp](#).

25.233.3.2 int done

When non-zero, this global means the user is done using this program.

Definition at line 80 of file [readline_autocomp.hpp](#).

25.233.3.3 char syscom[1024] [static]

String to pass to system(). This is for the LIST, VIEW and RENAME commands.

Definition at line 259 of file [readline_autocomp.hpp](#).

25.234 readline_autocomp.hpp

```

00001
00006 #ifndef __AIRINV_READLINE_AUTOCOMP_HPP
00007 #define __AIRINV_READLINE_AUTOCOMP_HPP
00008
00009 // STL
00010 #include <string>
00011 #include <iosfwd>
00012 #include <cstdio>
00013 #include <sys/types.h>
00014 #include <sys/file.h>
00015 #include <sys/stat.h>
00016 #include <sys/errno.h>
00017
00018 #include <readline/readline.h>
00019 #include <readline/history.h>
00020
00021 extern char* getwd();
00022 extern char* xmalloc (size_t);
00023
00024 /* The names of functions that actually do the manipulation. */
00025 int com_list (char*);
00026 int com_view (char*);
00027 int com_rename (char*);
00028 int com_stat (char*);
00029 int com_pwd (char*);
00030 int com_delete (char*);
00031 int com_help (char*);
00032 int com_cd (char*);
00033 int com_quit (char*);
00034
00035 typedef int (*pt2Func) (char*);
00036
00041 typedef struct {
00045     char const* name;
00046
00050     pt2Func *func;
00051
00055     char *doc;
00056 } COMMAND;
00057
00058 COMMAND commands[] = {
00059     { "cd", (*com_cd)(), "Change to directory DIR" },
00060     { "delete", com_delete, "Delete FILE" },
00061     { "help", com_help, "Display this text" },
00062     { "?", com_help, "Synonym for 'help'" },
00063     { "list", com_list, "List files in DIR" },
00064     { "ls", com_list, "Synonym for 'list'" },
00065     { "pwd", com_pwd, "Print the current working directory" },
00066     { "quit", com_quit, "Quit using airinv" },
00067     { "rename", com_rename, "Rename FILE to NEWNAME" },
00068     { "stat", com_stat, "Print out statistics on FILE" },
00069     { "view", com_view, "View the contents of FILE" },
00070     { (char*) NULL, (pt2Func) NULL, (char*) NULL }
00071 };
00072
00073 // Forward declarations
00074 char* stripwhite (char* iString);
00075 COMMAND* find_command (char* iString);
00076
00080 int done;
00081
00085 char* dupstr (char* iString) {
00086     char* r = xmalloc (std::strlen (iString) + 1);
00087     strcpy (r, iString);
00088     return r;
00089 }
00090

```



```

00094 int execute_line (char* line) {
00095     register int i;
00096     COMMAND* command;
00097     char* word;
00098
00099     /* Isolate the command word. */
00100     i = 0;
00101     while (line[i] && whitespace (line[i])) {
00102         i++;
00103     }
00104     word = line + i;
00105
00106     while (line[i] && !whitespace (line[i])) {
00107         i++;
00108     }
00109
00110     if (line[i]) {
00111         line[i++] = '\0';
00112     }
00113
00114     command = find_command (word);
00115
00116     if (!command) {
00117         std::cerr << word << ": No such command for airinv." << std::endl;
00118         return -1;
00119     }
00120
00121     /* Get argument to command, if any. */
00122     while (whitespace (line[i])) {
00123         i++;
00124     }
00125
00126     word = line + i;
00127
00128     /* Call the function. */
00129     return (*(command->func)) (word);
00130 }
00131
00136 COMMAND* find_command (char* name) {
00137     register int i;
00138
00139     for (i = 0; commands[i].name; i++) {
00140         if (strcmp (name, commands[i].name) == 0) {
00141             return (&commands[i]);
00142         }
00143     }
00144
00145     return (COMMAND*) NULL;
00146 }
00147
00152 char* stripwhite (char* string) {
00153     register char *s, *t;
00154
00155     for (s = string; whitespace (*s); s++) {
00156     }
00157
00158     if (*s == 0) {
00159         return s;
00160     }
00161
00162     t = s + strlen (s) - 1;
00163     while (t > s && whitespace (*t)) {
00164         t--;
00165     }
00166     *++t = '\0';
00167
00168     return s;
00169 }
00170
00171 /* ***** */
00172 /*
00173 /*             Interface to Readline Completion             */
00174 /*
00175 /* ***** */
00176
00177 char* command_generator (char* text, int state);
00178 char** fileman_completion (char* text, int start, int end);
00179
00185 void initialize_readline() {
00186     /* Allow conditional parsing of the ~/.inputrc file. */
00187     rl_readline_name = "airinv";
00188
00189     /* Tell the completer that we want a crack first. */
00190     rl_attempted_completion_function = (rl_completion_func_t*) fileman_completion
;
00191 }
00192

```

```

00200 char** fileman_completion (char* text, int start, int end) {
00201     char **matches;
00202
00203     matches = (char**) NULL;
00204
00210     if (start == 0) {
00211         matches = completion_matches (text, command_generator);
00212     }
00213
00214     return matches;
00215 }
00216
00222 char* command_generator (char* text, int state) {
00223     static int list_index, len;
00224     char* name;
00225
00231     if (!state) {
00232         list_index = 0;
00233         len = strlen (text);
00234     }
00235
00236     /* Return the next name which partially matches from the command list. */
00237     while (name = commands[list_index].name) {
00238         ++list_index;
00239
00240         if (strncmp (name, text, len) == 0) {
00241             return dupstr (name);
00242         }
00243     }
00244
00245     /* If no names matched, then return NULL. */
00246     return (char*) NULL;
00247 }
00248
00249 /* ***** */
00250 /* */
00251 /*             airinv Commands */
00252 /* */
00253 /* ***** */
00254
00259 static char syscom[1024];
00260
00264 void com_list (char* arg) {
00265     if (!arg) {
00266         arg = "";
00267     }
00268
00269     std::ostringstream oStr;
00270     oStr << "ls -FClg " << arg;
00271     return system (oStr.c_str());
00272 }
00273
00274 int com_view (char* arg) {
00275     if (!valid_argument ("view", arg)) {
00276         return 1;
00277     }
00278
00279     std::ostringstream oStr;
00280     oStr << "more " << arg;
00281     return system (syscom);
00282 }
00283
00284 int com_rename (char* arg) {
00285     too_dangerous ("rename");
00286     return 1;
00287 }
00288
00289 int com_stat (char* arg) {
00290     struct stat finfo;
00291
00292     if (!valid_argument ("stat", arg)) {
00293         return 1;
00294     }
00295
00296     if (stat (arg, &finfo) == -1) {
00297         perror (arg);
00298         return 1;
00299     }
00300
00301     std::cout << "Statistics for '" << arg << "':" << std::endl;
00302
00303     const std::string lPluralEnd1 = (finfo.st_nlink == 1) ? "" : "s";
00304     const std::string lPluralEnd2 = (finfo.st_size == 1) ? "" : "s";
00305     std::cout << arg << " has "
00306               << finfo.st_nlink << " link" << lPluralEnd1 << ", and is "
00307               << finfo.st_size << " byte" << lPluralEnd2 << " in length."
00308               << std::endl;

```

```

00309     std::cout << " Inode Last Change at: " << ctime (&finfo.st_ctime) <<
std::endl;
00310     std::cout << " Last access at: " << ctime (&finfo.st_atime) << std::endl;
00311     std::cout << " Last modified at: " << ctime (&finfo.st_mtime) << std::endl;
00312     return 0;
00313 }
00314
00315 int com_delete (char* arg) {
00316     too_dangerous ("delete");
00317     return 1;
00318 }
00319
00324 int com_help (char* arg) {
00325     register int i;
00326     int printed = 0;
00327
00328     for (i = 0; commands[i].name; i++) {
00329         if (!*arg || (strcmp (arg, commands[i].name) == 0)) {
00330             printf ("%s\t\t%s.\n", commands[i].name, commands[i].doc);
00331             printed++;
00332         }
00333     }
00334
00335     if (!printed) {
00336         printf ("No commands match '%s'. Possibilities are:\n", arg);
00337
00338         for (i = 0; commands[i].name; i++) {
00339             /* Print in six columns. */
00340             if (printed == 6) {
00341                 printed = 0;
00342                 printf ("\n");
00343             }
00344
00345             printf ("%s\t", commands[i].name);
00346             printed++;
00347         }
00348
00349         if (printed)
00350             printf ("\n");
00351     }
00352     return 0;
00353 }
00354
00355 /* Change to the directory ARG. */
00356 int com_cd (char* arg) {
00357     if (chdir (arg) == -1) {
00358         perror (arg);
00359         return 1;
00360     }
00361
00362     com_pwd ("");
00363     return 0;
00364 }
00365
00366 /* Print out the current working directory. */
00367 int com_pwd (char* ignore) {
00368     char dir[1024], *s;
00369
00370     s = getwd (dir);
00371     if (s == 0) {
00372         printf ("Error getting pwd: %s\n", dir);
00373         return 1;
00374     }
00375
00376     printf ("Current directory is %s\n", dir);
00377     return 0;
00378 }
00379
00380 /* The user wishes to quit using this program. Just set DONE non-zero. */
00381 int com_quit (char* arg) {
00382     done = 1;
00383     return 0;
00384 }
00385
00386 /* Function which tells you that you can't do this. */
00387 void too_dangerous (char* caller) {
00388     fprintf (stderr,
00389             "%s: Too dangerous for me to distribute. Write it yourself.\n",
00390             caller);
00391 }
00392
00393 /* Return non-zero if ARG is a valid argument for CALLER, else print
00394  * an error message and return zero. */
00395 int valid_argument (char* caller, char* arg) {
00396     if (!arg || !*arg) {
00397         fprintf (stderr, "%s: Argument required.\n", caller);
00398         return 0;

```

```

00399     }
00400
00401     return 1;
00402 }
00403
00404 #endif // _AIRINV_READLINE_AUTOCOMP_HPP

```

25.235 airinv/ui/cmdline/SReadline.hpp File Reference

C++ wrapper around libreadline.

```

#include <cstdio>
#include <readline/readline.h>
#include <readline/history.h>
#include <readline/keymaps.h>
#include <string>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <map>
#include <boost/algorithm/string/trim.hpp>
#include <boost/tokenizer.hpp>
#include <boost/function.hpp>

```

Classes

- class [swift::SKeymap](#)
The readline keymap wrapper.
- class [swift::SReadline](#)
The readline library wrapper.

Namespaces

- namespace [swift](#)
The wrapper namespace.

25.235.1 Detailed Description

C++ wrapper around libreadline. Supported: editing, history, custom completers, keymaps. Attention: implementation is not thread safe! It is mainly because the readline library provides pure C interface and has many calls for an "atomic" completion operation

Definition in file [SReadline.hpp](#).

25.236 SReadline.hpp

```

00001
00011 //
00012 // Date:      17 December 2005
00013 //           03 April    2006
00014 //           20 April    2006
00015 //           07 May      2006
00016 //
00017 // Copyright (c) Sergey Satskiy 2005 - 2006
00018 //           <sergesatsky@yahoo.com>
00019 //
00020 // Permission to copy, use, modify, sell and distribute this software
00021 // is granted provided this copyright notice appears in all copies.
00022 // This software is provided "as is" without express or implied
00023 // warranty, and with no claim as to its suitability for any purpose.

```

```

00024 //
00025
00026 #ifndef SREADLINE_H
00027 #define SREADLINE_H
00028
00029 #include <cstdio>
00030
00031 #include <readline/readline.h>
00032 #include <readline/history.h>
00033 #include <readline/keymaps.h>
00034
00035 #include <string>
00036 #include <fstream>
00037 #include <vector>
00038 #include <stdexcept>
00039 #include <map>
00040
00041 #include <boost/algorithm/string/trim.hpp>
00042 #include <boost/tokenizer.hpp>
00043 #include <boost/function.hpp>
00044
00045 namespace {
00054     typedef std::vector<std::string> TokensStorage;
00055
00059     typedef std::vector<TokensStorage> CompletionsStorage;
00060
00064     typedef boost::function<int (int, int)> KeyCallback;
00065
00069     typedef std::map<int, KeyCallback> KeysBind;
00070
00074     const size_t DefaultHistoryLimit (64);
00075
00079     CompletionsStorage Completions;
00080
00084     TokensStorage Tokens;
00085
00089     std::map<Keymap, KeysBind> Keymaps;
00090
00094     bool KeymapWasSetup (false);
00095
00099     Keymap Earlykeymap (0);
00100
00101
00108     char* Generator (const char* text, int State);
00109
00110
00118     char** UserCompletion (const char* text, int start, int end);
00119
00120
00128     int KeyDispatcher (int Count, int Key);
00129
00130
00135     int StartupHook (void);
00136
00137
00145     template <typename Container>
00146     bool AreTokensEqual (const Container& Pattern, const Container& Input) {
00147         if (Input.size() > Pattern.size()) {
00148             return false;
00149         }
00150
00151         typename Container::const_iterator k (Pattern.begin());
00152         typename Container::const_iterator j (Input.begin());
00153         for (; j != Input.end(); ++k, ++j) {
00154             const std::string lPattern = *k;
00155             if (lPattern == "%file") {
00156                 continue;
00157             }
00158
00159             const std::string lInput = *j;
00160             if (lPattern != lInput) {
00161                 return false;
00162             }
00163         }
00164         return true;
00165     }
00166
00167     // See description near the prototype
00168     template <typename ContainerType>
00169     void SplitTokens (const std::string& Source, ContainerType& Container) {
00170         typedef boost::tokenizer<boost::char_separator<char> > TokenizerType;
00171
00172         // Set of token separators
00173         boost::char_separator<char> Separators (" \\t\\n");
00174         // Tokens provider
00175         TokenizerType Tokenizer (Source, Separators);

```

```

00176
00177     Container.clear();
00178     for (TokenizerType::const_iterator k (Tokenizer.begin());
00179         k != Tokenizer.end(); ++k) {
00180         // Temporary storage for the token, in order to trim that latter
00181         std::string SingleToken (*k);
00182
00183         boost::algorithm::trim (SingleToken);
00184         Container.push_back (SingleToken);
00185     }
00186 }
00187
00188 // See description near the prototype
00189 char** UserCompletion (const char* text, int start, int end) {
00190     // No default completion at all
00191     rl_attempted_completion_over = 1;
00192
00193     if (Completions.empty() == true) {
00194         return NULL;
00195     }
00196
00197     // Memorise all the previous tokens
00198     std::string PreInput (rl_line_buffer, start);
00199     SplitTokens (PreInput, Tokens);
00200
00201     // Detect whether we should call the standard file name completer
00202     // or a custom one
00203     bool FoundPretender (false);
00204
00205     for (CompletionsStorage::const_iterator k (Completions.begin());
00206         k != Completions.end(); ++k) {
00207         const TokensStorage& lTokenStorage = *k;
00208         if (AreTokensEqual (lTokenStorage, Tokens) == false) {
00209             continue;
00210         }
00211
00212         if (lTokenStorage.size() > Tokens.size()) {
00213             FoundPretender = true;
00214             if (lTokenStorage [Tokens.size()] == "%file") {
00215                 // Standard file name completer - called for the "%file" keyword
00216                 return rl_completion_matches (text, rl_filename_completion_function);
00217             }
00218         }
00219     }
00220
00221     if (FoundPretender) {
00222         return rl_completion_matches (text, Generator);
00223     }
00224     return NULL;
00225 }
00226
00227 // See description near the prototype
00228 char* Generator (const char* text, int State) {
00229     static int Length;
00230     static CompletionsStorage::const_iterator Iterator;
00231
00232     if ( State == 0 ) {
00233         Iterator = Completions.begin();
00234         Length = strlen (text);
00235     }
00236
00237     for ( ; Iterator != Completions.end(); ++Iterator) {
00238         const TokensStorage& lCompletion = *Iterator;
00239         if (AreTokensEqual (lCompletion, Tokens) == false) {
00240             continue;
00241         }
00242
00243         if (lCompletion.size() > Tokens.size()) {
00244             if (lCompletion [Tokens.size()] == "%file") {
00245                 continue;
00246             }
00247
00248             const char* lCompletionCharStr (lCompletion [Tokens.size()].c_str());
00249             if (strcmp (text, lCompletionCharStr, Length) == 0) {
00250                 // Readline will free the allocated memory
00251                 const size_t lCompletionSize = strlen (lCompletionCharStr) + 1;
00252                 char* NewString (static_cast<char*> (malloc (lCompletionSize)));
00253                 strcpy (NewString, lCompletionCharStr);
00254
00255                 ++Iterator;
00256
00257                 return NewString;
00258             }
00259         }
00260     }
00261
00262     return NULL;

```

```

00263     }
00264
00265
00266     // See the description near the prototype
00267     int KeyDispatcher (int Count, int Key ) {
00268         std::map< Keymap, KeysBind >::iterator Set (Keymaps.find (rl_get_keymap()))
;
00269         if (Set == Keymaps.end()) {
00270             // Most probably it happens because the header was
00271             // included into many compilation units and the
00272             // keymap setting calls were made in different files.
00273             // This is the problem of "global" data.
00274             // The storage of all the registered keymaps is in anonymous
00275             // namespace.
00276             throw std::runtime_error ("Error selecting a keymap.");
00277         }
00278
00279         (Set->second)[Key] (Count, Key);
00280         return 0;
00281     }
00282
00283     // See the description near the prototype
00284     int StartupHook (void) {
00285         if (KeymapWasSetup) {
00286             rl_set_keymap (Earlykeymap);
00287         }
00288         return 0;
00289     }
00290 } // Anonymous namespace
00291
00292
00293 namespace swift {
00294
00300     class SKeymap {
00301     private:
00302         // Readline keymap
00303         Keymap keymap;
00304
00305     public:
00306         explicit SKeymap (bool PrintableBound = false) : keymap (NULL) {
00307             if (PrintableBound == true) {
00308                 // Printable characters are bound
00309                 keymap = rl_make_keymap();
00310             } else {
00311                 // Empty keymap
00312                 keymap = rl_make_bare_keymap();
00313             }
00314
00315             if (keymap == NULL) {
00316                 throw std::runtime_error ("Cannot allocate keymap.");
00317             }
00318
00319             // Register a new keymap in the global list
00320             Keymaps [keymap] = KeysBind();
00321         }
00322
00323         explicit SKeymap (Keymap Pattern) : keymap (rl_copy_keymap (Pattern)) {
00324             if ( keymap == NULL ) {
00325                 throw std::runtime_error( "Cannot allocate keymap." );
00326             }
00327
00328             // Register a new keymap in the global list
00329             Keymaps [keymap] = KeysBind();
00330         }
00331
00332         ~SKeymap() {
00333             // Deregister the keymap
00334             Keymaps.erase (keymap);
00335             rl_discard_keymap (keymap);
00336         }
00337
00338         void Bind (int Key, KeyCallback Callback) {
00339             Keymaps [keymap][Key] = Callback;
00340
00341             if (rl_bind_key_in_map (Key, KeyDispatcher, keymap) != 0) {
00342                 // Remove from the map just bound key
00343                 Keymaps [keymap].erase (Key);
00344                 throw std::runtime_error ("Invalid key.");
00345             }
00346         }
00347
00348         void Unbind (int Key) {
00349             rl_unbind_key_in_map (Key, keymap);
00350             Keymaps [keymap].erase (Key);
00351         }
00352     }

```

```

00385
00386 // void Bind (const std::string& Sequence, boost::function<int (int,
// void Unbind (std::string& Sequence);
00387
00388
00389 public:
00395 SKeymap (const SKeymap& rhs) {
00396     if (this == &rhs) {
00397         return;
00398     }
00399     keymap = rl_copy_keymap (rhs.keymap);
00400 }
00401
00407 SKeymap& operator= (const SKeymap& rhs) {
00408     if (this == &rhs) {
00409         return *this;
00410     }
00411     keymap = rl_copy_keymap (rhs.keymap);
00412     return *this;
00413 }
00414
00415 friend class SReadline;
00416 };
00417
00424 class SReadline {
00425 public:
00431 SReadline (const size_t Limit = DefaultHistoryLimit)
00432     : HistoryLimit (Limit), HistoryFileName (""),
00433       OriginalCompletion (rl_attempted_completion_function) {
00434     rl_startup_hook = StartupHook;
00435     rl_attempted_completion_function = UserCompletion;
00436     using_history();
00437 }
00438
00446 SReadline (const std::string& historyFileName,
00447             const size_t Limit = DefaultHistoryLimit)
00448     : HistoryLimit (Limit), HistoryFileName (historyFileName),
00449       OriginalCompletion (rl_attempted_completion_function) {
00450     rl_startup_hook = StartupHook;
00451     rl_attempted_completion_function = UserCompletion;
00452     using_history();
00453     LoadHistory (HistoryFileName);
00454 }
00455
00460 ~SReadline() {
00461     rl_attempted_completion_function = OriginalCompletion;
00462     SaveHistory (HistoryFileName);
00463 }
00464
00471 std::string GetLine (const std::string& Prompt) {
00472     bool Unused;
00473     return GetLine (Prompt, Unused);
00474 }
00475
00484 template <typename Container>
00485 std::string GetLine (const std::string& Prompt, Container& ReadTokens) {
00486     bool Unused;
00487     return GetLine (Prompt, ReadTokens, Unused);
00488 }
00489
00499 template <typename Container>
00500 std::string GetLine (const std::string& Prompt, Container& ReadTokens,
00501                     bool& BreakOut) {
00502     std::string Input (GetLine (Prompt, BreakOut));
00503     SplitTokens (Input, ReadTokens);
00504     return Input;
00505 }
00506
00507
00515 std::string GetLine (const std::string& Prompt, bool& BreakOut) {
00516     BreakOut = true;
00517
00518     char* ReadLine (getline (Prompt.c_str()));
00519     if (ReadLine == NULL) {
00520         return std::string();
00521     }
00522
00523     // It's OK
00524     BreakOut = false;
00525     std::string Input (ReadLine);
00526     free (ReadLine); ReadLine = NULL;
00527
00528     boost::algorithm::trim (Input);
00529     if (Input.empty() == false) {
00530         if (history_length == 0
00531             || Input != history_list()[ history_length - 1 ]->line) {
00531             add_history (Input.c_str());
00532         }
00533     }

```



```

00533
00534         if (history_length >= static_cast<int> (HistoryLimit)) {
00535             stifile_history (HistoryLimit);
00536         }
00537     }
00538 }
00539
00540     return Input;
00541 }
00542
00543
00544 template <typename ContainerType>
00550 void GetHistory (ContainerType& Container) {
00551     for (int k (0); k < history_length; ++k ) {
00552         Container.push_back (history_list()[k]->line);
00553     }
00554 }
00555
00562 bool SaveHistory (std::ostream& OS) {
00563     if (!OS) {
00564         return false;
00565     }
00566
00567     for (int k (0); k < history_length; ++k) {
00568         OS << history_list()[ k ]->line << std::endl;
00569     }
00570     return true;
00571 }
00572
00579 bool SaveHistory (const std::string& FileName) {
00580     if (FileName.empty() == true) {
00581         return false;
00582     }
00583
00584     std::ofstream OS (FileName.c_str());
00585     return SaveHistory (OS);
00586 }
00587
00592 void ClearHistory() {
00593     clear_history();
00594 }
00595
00602 bool LoadHistory (std::istream& IS) {
00603     if (!IS) {
00604         return false;
00605     }
00606
00607     ClearHistory();
00608     std::string OneLine;
00609
00610     while (!getline (IS, OneLine).eof()) {
00611         boost::algorithm::trim( OneLine );
00612         if ((history_length == 0)
00613             || OneLine != history_list()[history_length - 1]->line) {
00614             add_history (OneLine.c_str());
00615         }
00616     }
00617     stifile_history (HistoryLimit);
00618     return true;
00619 }
00620
00627 bool LoadHistory (const std::string& FileName) {
00628     if (FileName.empty() == true) {
00629         return false;
00630     }
00631
00632     std::ifstream IS (FileName.c_str());
00633     return LoadHistory (IS);
00634 }
00635
00655 template <typename ContainerType>
00656 void RegisterCompletions (const ContainerType& Container) {
00657     Completions.clear();
00658     for (typename ContainerType::const_iterator k (Container.begin());
00659          k != Container.end(); ++k) {
00660         std::vector<std::string> OneLine;
00661         const std::string& kStr = static_cast<std::string> (*k);
00662         SplitTokens (kStr, OneLine);
00663         Completions.push_back (OneLine);
00664     }
00665 }
00666
00673 void SetKeymap (SKeymap& NewKeymap) {
00674     rl_set_keymap (NewKeymap.keymap);
00675     KeymapWasSetup = true;
00676     Earlykeymap = NewKeymap.keymap;

```

```

00677     }
00678
00679
00680 private:
00681     // ////////////////////////////////// Attributes //////////////////////////////////
00682     const size_t HistoryLimit;
00683
00684     const std::string HistoryFileName;
00685
00686     rl_completion_func_t* OriginalCompletion;
00687 };
00688 }; // namespace swift
00689
00690 #endif
00691
00692

```

25.237 doc/local/authors.doc File Reference

25.238 doc/local/codingrules.doc File Reference

25.239 doc/local/copyright.doc File Reference

25.240 doc/local/documentation.doc File Reference

25.241 doc/local/features.doc File Reference

25.242 doc/local/help_wanted.doc File Reference

25.243 doc/local/howto_release.doc File Reference

25.244 doc/local/index.doc File Reference

25.245 doc/local/installation.doc File Reference

25.246 doc/local/linking.doc File Reference

25.247 doc/local/test.doc File Reference

25.248 doc/local/users_guide.doc File Reference

25.249 doc/local/verification.doc File Reference

25.250 doc/tutorial/tutorial.doc File Reference

25.251 test/airinv/InventoryTestSuite.cpp File Reference

25.252 InventoryTestSuite.cpp

```

00001
00002 // //////////////////////////////////
00003 // Import section
00004 // //////////////////////////////////
00005 // STL
00006 #include <sstream>
00007 #include <fstream>
00008 #include <string>
00009 // Boost Unit Test Framework (UTF)
00010 #define BOOST_TEST_DYN_LINK
00011 #define BOOST_TEST_MAIN
00012 #define BOOST_TEST_MODULE InventoryTestSuite
00013 #include <boost/test/unit_test.hpp>
00014 // StdAir
00015 #include <stdair/basic/BasLogParams.hpp>
00016

```

```

00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/service/Logger.hpp>
00024 #include <stdair/stdair_exceptions.hpp>
00025 // Airinv
00026 #include <airinv/AIRINV_Types.hpp>
00027 #include <airinv/AIRINV_Master_Service.hpp>
00028 #include <airinv/config/airinv-paths.hpp>
00029
00030 namespace boost_utf = boost::unit_test;
00031
00032 // (Boost) Unit Test XML Report
00033 std::ofstream utfReportStream ("InventoryTestSuite_utfresults.xml");
00034
00035 struct UnitTestConfig {
00036     UnitTestConfig() {
00037         boost_utf::unit_test_log.set_stream (utfReportStream);
00038         boost_utf::unit_test_log.set_format (boost_utf::XML);
00039         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00040         //boost_utf::unit_test_log.set_threshold_level
00041         (boost_utf::log_successful_tests);
00042     }
00043
00044     ~UnitTestConfig() {
00045     }
00046 };
00047
00048 // //////////////////////////////////////
00049 bool testInventoryHelper (const unsigned short iTestFlag,
00050                          const stdair::Filename_T& iInventoryInputFilename,
00051                          const stdair::Filename_T& iScheduleInputFilename,
00052                          const stdair::Filename_T& iODInputFilename,
00053                          const stdair::Filename_T& iYieldInputFilename,
00054                          const bool isBuiltin,
00055                          const bool isForSchedule) {
00056
00057     // Output log File
00058     std::ostringstream oStr;
00059     oStr << "InventoryTestSuite_" << iTestFlag << ".log";
00060     const stdair::Filename_T lLogFilename (oStr.str());
00061
00062     // Set the log parameters
00063     std::ofstream logOutputFile;
00064     // Open and clean the log outputfile
00065     logOutputFile.open (lLogFilename.c_str());
00066     logOutputFile.clear();
00067
00068     // Initialise the AirInv service object
00069     const bool lForceMultipleInit = true;
00070     stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00071                                     logOutputFile,
00072                                     lForceMultipleInit);
00073
00074     // Initialise the inventory service
00075     AIRINV::AIRINV_Master_Service airinvService (lLogParams);
00076
00077     // Parameters for the sale
00078     std::string lSegmentDateKey;
00079     stdair::ClassCode_T lClassCode;
00080     const stdair::PartySize_T lPartySize (2);
00081
00082     // Check whether or not a (CSV) input file should be read
00083     if (isBuiltin == true) {
00084         // Build the default sample BOM tree (filled with inventories) for AirInv
00085         airinvService.buildSampleBom();
00086
00087         // Define a specific segment-date key for the sample BOM tree
00088         lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00089         lClassCode = "Q";
00090     } else {
00091         if (isForSchedule == true) {
00092             // Build the BOM tree from parsing a schedule file (and O&D list)
00093             AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
00094             airinvService.parseAndLoad (iScheduleInputFilename, iODInputFilename,
00095                                       lYieldFilePath);
00096
00097             // Define a specific segment-date key for the schedule-based inventory
00098             lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00099             lClassCode = "Y";
00100         } else {
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112

```

```

00113         // Build the BOM tree from parsing an inventory dump file
00114         airinvService.parseAndLoad (iInventoryInputFilename);
00115
00116         // Define a specific segment-date key for the inventory parsed file
00117         //const std::string lSegmentDateKey ("SV, 5, 2010-03-11, KBP, JFK,
00118         08:00:00");
00119         lSegmentDateKey = "SV, 5, 2010-03-11, KBP, JFK, 08:00:00";
00120         lClassCode = "J";
00121     }
00122 }
00123
00124 // Make a booking
00125 const bool hasSaleBeenSuccessful =
00126     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00127
00128 // DEBUG: Display the list of travel solutions
00129 const std::string& lCSVDump = airinvService.csvDisplay();
00130 STDAIR_LOG_DEBUG (lCSVDump);
00131
00132 // Close the log file
00133 logOutputFile.close();
00134
00135 if (hasSaleBeenSuccessful == false) {
00136     STDAIR_LOG_DEBUG ("No sale can be made for '" << lSegmentDateKey
00137         << "'");
00138 }
00139
00140 return hasSaleBeenSuccessful;
00141
00142 }
00143
00144 // //////////// Main: Unit Test Suite ////////////
00145
00146 // Set the UTF configuration (re-direct the output to a specific file)
00147 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00148
00149 // Start the test suite
00150 BOOST_AUTO_TEST_SUITE (master_test_suite)
00151
00152 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell) {
00153     // Input file name
00154     const stdair::Filename_T lInventoryInputFilename (STDAIR_SAMPLE_DIR
00155         "/invdump01.csv");
00156
00157     // State whether the BOM tree should be built-in or parsed from an input file
00158     const bool isBuiltin = false;
00159     // State whether the BOM tree should be built from a schedule file (instead
00160     of from an inventory dump)
00161     const bool isForSchedule = false;
00162
00163     // Try sell a default segment.
00164     bool hasTestBeenSuccessful = false;
00165     BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00166         testInventoryHelper (0, lInventoryInputFilename,
00167             " ", " ", " ", " ", isBuiltin,
00168             isForSchedule));
00169     BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00170 }
00171
00172
00173 }
00174
00175 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_built_in) {
00176     // State whether the BOM tree should be built-in or parsed from an input file
00177     const bool isBuiltin = true;
00178     // State whether the BOM tree should be built from a schedule file (instead
00179     of from an inventory dump)
00180     const bool isForSchedule = false;
00181
00182     // Try sell a default segment.
00183     bool hasTestBeenSuccessful = false;
00184     BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00185         testInventoryHelper (1, " ", " ", " ", " ", " ",
00186             isBuiltin, isForSchedule));
00187     BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00188 }
00189
00190 }
00191
00192 }
00193
00194 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_schedule) {
00195     // Input file names
00196     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00197         "/schedule01.csv");
00198     const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00199         "/ond01.csv");

```

```

00204     const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00205                                                    "/yieldstore01.csv");
00206
00207     // State whether the BOM tree should be built-in or parsed from an input file
00208     const bool isBuiltin = false;
00209     // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00210     const bool isForSchedule = true;
00211
00212     // Try sell a default segment.
00213     bool hasTestBeenSuccessful = false;
00214     BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00215                           testInventoryHelper (2, " ",
00216                                                lScheduleInputFilename,
00217                                                lODInputFilename,
00218                                                lYieldInputFilename,
00219                                                isBuiltin, isForSchedule));
00220     BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00221 }
00222 }
00223
00228 BOOST_AUTO_TEST_CASE (airinv_error_inventory_input_file) {
00229
00230     // Inventory input file name
00231     const stdair::Filename_T lMissingInventoryFilename (STDAIR_SAMPLE_DIR
00232                                                         "/missingFile.csv");
00233
00234     // State whether the BOM tree should be built-in or parsed from an input file
00235     const bool isBuiltin = false;
00236     // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00237     const bool isForSchedule = false;
00238
00239     // Try sell a default segment.
00240     BOOST_CHECK_THROW (testInventoryHelper (3, lMissingInventoryFilename,
00241                                             " ", " ", " ", isBuiltin,
isForSchedule),
00242                       AIRINV::InventoryInputFileNotFoundException);
00243 }
00244 }
00245
00250 BOOST_AUTO_TEST_CASE (airinv_error_schedule_input_file) {
00251
00252     // Schedule input file name
00253     const stdair::Filename_T lMissingScheduleFilename (STDAIR_SAMPLE_DIR
00254                                                         "/missingFile.csv");
00255
00256     // State whether the BOM tree should be built-in or parsed from an input file
00257     const bool isBuiltin = false;
00258     // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00259     const bool isForSchedule = true;
00260
00261     // Try sell a default segment.
00262     BOOST_CHECK_THROW (testInventoryHelper (4, " ", lMissingScheduleFilename,
00263                                             " ", " ", isBuiltin, isForSchedule),
00264                       AIRINV::ScheduleInputFileNotFoundException);
00265 }
00266 }
00267
00272 BOOST_AUTO_TEST_CASE (airinv_error_yield_input_file) {
00273
00274     // Input file names
00275     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00276                                                         "/schedule01.csv");
00277     const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00278                                                  "/ond01.csv");
00279     const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00280                                                    "/missingFile.csv");
00281
00282     // State whether the BOM tree should be built-in or parsed from an input file
00283     const bool isBuiltin = false;
00284     // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00285     const bool isForSchedule = true;
00286
00287     // Try sell a default segment.
00288     BOOST_CHECK_THROW (testInventoryHelper (5, " ",
00289                                             lScheduleInputFilename,
00290                                             lODInputFilename,
00291                                             lYieldInputFilename,
00292                                             isBuiltin, isForSchedule),
00293                       AIRINV::YieldInputFileNotFoundException);
00294 }
00295 }
00296
00301 BOOST_AUTO_TEST_CASE (airinv_error_flight_date_duplication) {

```

```

00302
00303 // Input file names
00304 const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00305 "/scheduleError01.csv");
00306 const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00307 "/ond01.csv");
00308 const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00309 "/missingFile.csv");
00310
00311 // State whether the BOM tree should be built-in or parsed from an input file
00312 const bool isBuiltin = false;
00313 // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00314 const bool isForSchedule = true;
00315
00316 // Try sell a default segment.
00317 BOOST_CHECK_THROW (testInventoryHelper (6, " ",
00318 lScheduleInputFilename,
00319 lODInputFilename,
00320 lYieldInputFilename,
00321 isBuiltin, isForSchedule),
00322 AIRINV::FlightDateDuplicationException);
00323
00324 }
00325
00330 BOOST_AUTO_TEST_CASE (airinv_error_schedule_parsing_failed) {
00331
00332 // Input file names
00333 const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00334 "/scheduleError02.csv");
00335 const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00336 "/ond01.csv");
00337 const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00338 "/yieldstore01.csv");
00339
00340 // State whether the BOM tree should be built-in or parsed from an input file
00341 const bool isBuiltin = false;
00342 // State whether the BOM tree should be built from a schedule file (instead
of from an inventory dump)
00343 const bool isForSchedule = true;
00344
00345 // Try sell a default segment.
00346 BOOST_CHECK_THROW (testInventoryHelper (7, " ",
00347 lScheduleInputFilename,
00348 lODInputFilename,
00349 lYieldInputFilename,
00350 isBuiltin, isForSchedule),
00351 AIRINV::ScheduleFileParsingFailedException);
00352
00353 }
00354
00355 // End the test suite
00356 BOOST_AUTO_TEST_SUITE_END ()
00357
00358

```

25.253 test/airinv/InventoryTestSuite.hpp File Reference

```

#include <iosfwd>
#include <cppunit/extensions/HelperMacros.h>

```

Classes

- class [InventoryTestSuite](#)

Functions

- [CPPUNIT_TEST_SUITE_REGISTRATION](#) ([InventoryTestSuite](#))

25.253.1 Function Documentation

25.253.1.1 CPPUNIT_TEST_SUITE_REGISTRATION ([InventoryTestSuite](#))

25.254 InventoryTestSuite.hpp

```
00001 // STL
00002 #include <iosfwd>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00007 class InventoryTestSuite : public CppUnit::TestFixture {
00008     CPPUNIT_TEST_SUITE (InventoryTestSuite);
00009     CPPUNIT_TEST (simpleInventory);
00010     // CPPUNIT_TEST (errorCase);
00011     CPPUNIT_TEST_SUITE_END ();
00012 public:
00013
00015     void simpleInventory();
00016
00018     // void errorCase ();
00019
00021     InventoryTestSuite ();
00022
00023 private:
00025     void simpleInventoryHelper();
00026
00027 protected:
00028     std::stringstream _describeKey;
00029 };
00030
00031 CPPUNIT_TEST_SUITE_REGISTRATION (InventoryTestSuite);
```

Index

- ~AIRINV_Service
 - AIRINV::AIRINV_Service, [125](#)
- ~AirInvServer
 - AIRINV::AirInvServer, [130](#)
- ~BomAbstract
 - AIRINV::BomAbstract, [131](#)
- ~FacAirinvMasterServiceContext
 - AIRINV::FacAirinvMasterServiceContext, [171](#)
- ~FacAirinvServiceContext
 - AIRINV::FacAirinvServiceContext, [172](#)
- ~FacBomAbstract
 - AIRINV::FacBomAbstract, [174](#)
- ~FacServiceAbstract
 - AIRINV::FacServiceAbstract, [176](#)
- ~FacSupervisor
 - AIRINV::FacSupervisor, [178](#)
- ~SKeymap
 - swift::SKeymap, [240](#)
- ~SReadline
 - swift::SReadline, [243](#)
- ~ServiceAbstract
 - AIRINV::ServiceAbstract, [239](#)
- _DCP
 - AIRINV::DCPEventStruct, [148](#)
- _DCPRule
 - AIRINV::DCPParserHelper::DCPRuleParser, [156](#)
 - AIRINV::DCPParserHelper::doEndDCP, [166](#)
 - AIRINV::DCPParserHelper::ParserSemanticAction, [224](#)
 - AIRINV::DCPParserHelper::storeAdvancePurchase, [249](#)
 - AIRINV::DCPParserHelper::storeAirlineCode, [253](#)
 - AIRINV::DCPParserHelper::storeCabinCode, [263](#)
 - AIRINV::DCPParserHelper::storeChangeFees, [266](#)
 - AIRINV::DCPParserHelper::storeChannel, [267](#)
 - AIRINV::DCPParserHelper::storeClass, [268](#)
 - AIRINV::DCPParserHelper::storeDateRangeEnd, [277](#)
 - AIRINV::DCPParserHelper::storeDateRangeStart, [281](#)
 - AIRINV::DCPParserHelper::storeDCP, [282](#)
 - AIRINV::DCPParserHelper::storeDCPId, [284](#)
 - AIRINV::DCPParserHelper::storeDestination, [285](#)
 - AIRINV::DCPParserHelper::storeEndRangeTime, [289](#)
 - AIRINV::DCPParserHelper::storeMinimumStay, [315](#)
 - AIRINV::DCPParserHelper::storeNonRefundable, [328](#)
 - AIRINV::DCPParserHelper::storeOrigin, [335](#)
 - AIRINV::DCPParserHelper::storePOS, [341](#)
 - AIRINV::DCPParserHelper::storeSaturdayStay, [347](#)
 - AIRINV::DCPParserHelper::storeStartRangeTime, [364](#)
- _acp
 - AIRINV::LegCabinStruct, [215](#)
- _adjustment
 - AIRINV::LegCabinStruct, [214](#)
- _advancePurchase
 - AIRINV::DCPEventStruct, [148](#)
- _airlineCode
 - AIRINV::DCPEventStruct, [148](#)
 - AIRINV::FlightDateStruct, [185](#)
 - AIRINV::FlightPeriodStruct, [194](#)
 - AIRINV::Request, [227](#)
 - stdair::BomPropertyTree, [133](#)
- _airlineCodeList
 - AIRINV::DCPEventStruct, [148](#)
- _airportCodeList
 - stdair::BomPropertyTree, [133](#)
- _airportList
 - AIRINV::FlightDateStruct, [187](#)
 - AIRINV::FlightPeriodStruct, [195](#)
- _airportOrderedList
 - AIRINV::FlightDateStruct, [187](#)
 - AIRINV::FlightPeriodStruct, [196](#)
- _areSegmentDefinitionsSpecific
 - AIRINV::FlightDateStruct, [188](#)
 - AIRINV::FlightPeriodStruct, [196](#)
- _au
 - AIRINV::LegCabinStruct, [215](#)
- _avPool
 - AIRINV::LegCabinStruct, [215](#)
- _availability
 - AIRINV::BucketStruct, [139](#)
- _boardingDate
 - AIRINV::LegStruct, [217](#)
 - AIRINV::SegmentStruct, [237](#)
- _boardingDateOffset
 - AIRINV::LegStruct, [217](#)
- _boardingPoint
 - AIRINV::LegStruct, [217](#)
 - AIRINV::SegmentStruct, [237](#)
- _boardingTime
 - AIRINV::LegStruct, [217](#)
 - AIRINV::SegmentStruct, [238](#)
- _bomRoot
 - AIRINV::DCPParserHelper::DCPRuleParser, [156](#)
 - AIRINV::DCPParserHelper::doEndDCP, [166](#)
 - AIRINV::InventoryParserHelper::doEndFlightDate, [169](#)
 - AIRINV::InventoryParserHelper::InventoryParser, [212](#)
 - AIRINV::ScheduleParserHelper::doEndFlight, [167](#)
 - AIRINV::ScheduleParserHelper::FlightPeriodParser, [190](#)
- _bucketList
 - AIRINV::LegCabinStruct, [216](#)
- _cabinCode

- AIRINV::DCPEventStruct, [147](#)
- AIRINV::LegCabinStruct, [214](#)
- AIRINV::SegmentCabinStruct, [234](#)
- _cabinList
 - AIRINV::LegStruct, [218](#)
 - AIRINV::SegmentStruct, [238](#)
- _changeFees
 - AIRINV::DCPEventStruct, [148](#)
- _channel
 - AIRINV::DCPEventStruct, [147](#)
- _classCode
 - AIRINV::BookingClassStruct, [135](#)
 - AIRINV::DCPEventStruct, [148](#)
- _classCodeList
 - AIRINV::DCPEventStruct, [149](#)
- _classList
 - AIRINV::FareFamilyStruct, [180](#)
- _classes
 - AIRINV::FareFamilyStruct, [180](#)
- _cumulatedProtection
 - AIRINV::BookingClassStruct, [136](#)
- _dateOffSet
 - AIRINV::FlightDateStruct, [186](#)
- _dateOffset
 - AIRINV::FlightPeriodStruct, [195](#)
- _dateRange
 - AIRINV::FlightPeriodStruct, [194](#)
- _dateRangeEnd
 - AIRINV::DCPEventStruct, [147](#)
 - AIRINV::FlightPeriodStruct, [195](#)
- _dateRangeStart
 - AIRINV::DCPEventStruct, [147](#)
 - AIRINV::FlightPeriodStruct, [195](#)
- _dcsRegrade
 - AIRINV::LegCabinStruct, [214](#)
- _departureDate
 - AIRINV::Request, [227](#)
 - stdair::BomPropertyTree, [133](#)
- _describeKey
 - InventoryTestSuite, [213](#)
- _destination
 - AIRINV::DCPEventStruct, [147](#)
- _dow
 - AIRINV::FlightPeriodStruct, [194](#)
- _elapsed
 - AIRINV::LegStruct, [218](#)
 - AIRINV::SegmentStruct, [238](#)
- _etb
 - AIRINV::BookingClassStruct, [137](#)
 - AIRINV::LegCabinStruct, [215](#)
- _familyCode
 - AIRINV::FareFamilyStruct, [180](#)
- _fareFamilies
 - AIRINV::SegmentCabinStruct, [234](#)
- _flightDate
 - AIRINV::FlightDateStruct, [186](#)
 - AIRINV::InventoryParserHelper::doEndFlightDate, [169](#)
- AIRINV::InventoryParserHelper::InventoryParser, [212](#)
- AIRINV::InventoryParserHelper::ParserSemanticAction, [220](#)
- AIRINV::InventoryParserHelper::storeACP, [247](#)
- AIRINV::InventoryParserHelper::storeAirlineCode, [250](#)
- AIRINV::InventoryParserHelper::storeAU, [254](#)
- AIRINV::InventoryParserHelper::storeBoardingDate, [256](#)
- AIRINV::InventoryParserHelper::storeBoardingTime, [257](#)
- AIRINV::InventoryParserHelper::storeBookingCounter, [260](#)
- AIRINV::InventoryParserHelper::storeBucketAvailability, [262](#)
- AIRINV::InventoryParserHelper::storeClassAvailability, [270](#)
- AIRINV::InventoryParserHelper::storeClassCode, [271](#)
- AIRINV::InventoryParserHelper::storeClassETB, [274](#)
- AIRINV::InventoryParserHelper::storeCumulatedProtection, [276](#)
- AIRINV::InventoryParserHelper::storeETB, [290](#)
- AIRINV::InventoryParserHelper::storeFamilyCode, [292](#)
- AIRINV::InventoryParserHelper::storeFCClasses, [294](#)
- AIRINV::InventoryParserHelper::storeFlightDate, [297](#)
- AIRINV::InventoryParserHelper::storeFlightNumber, [299](#)
- AIRINV::InventoryParserHelper::storeFlightTypeCode, [302](#)
- AIRINV::InventoryParserHelper::storeFlightVisibilityCode, [303](#)
- AIRINV::InventoryParserHelper::storeGAV, [305](#)
- AIRINV::InventoryParserHelper::storeLegBoardingPoint, [307](#)
- AIRINV::InventoryParserHelper::storeLegCabinCode, [311](#)
- AIRINV::InventoryParserHelper::storeLegOffPoint, [314](#)
- AIRINV::InventoryParserHelper::storeNAV, [317](#)
- AIRINV::InventoryParserHelper::storeNbOfBkgs, [318](#)
- AIRINV::InventoryParserHelper::storeNbOfGroupBkgs, [320](#)
- AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs, [321](#)
- AIRINV::InventoryParserHelper::storeNbOfStaffBkgs, [323](#)
- AIRINV::InventoryParserHelper::storeNbOfWL-Bkgs, [324](#)
- AIRINV::InventoryParserHelper::storeNego, [326](#)
- AIRINV::InventoryParserHelper::storeNoShow, [329](#)

- AIRINV::InventoryParserHelper::storeOffDate, [330](#)
- AIRINV::InventoryParserHelper::storeOffTime, [332](#)
- AIRINV::InventoryParserHelper::storeOverbooking, [336](#)
- AIRINV::InventoryParserHelper::storeParentClass-Code, [338](#)
- AIRINV::InventoryParserHelper::storeParent-SubclassCode, [339](#)
- AIRINV::InventoryParserHelper::storeProtection, [342](#)
- AIRINV::InventoryParserHelper::storeRevenue-Availability, [344](#)
- AIRINV::InventoryParserHelper::storeSaleable-Capacity, [345](#)
- AIRINV::InventoryParserHelper::storeSeatIndex, [348](#)
- AIRINV::InventoryParserHelper::storeSegment-Availability, [350](#)
- AIRINV::InventoryParserHelper::storeSegment-BoardingPoint, [351](#)
- AIRINV::InventoryParserHelper::storeSegment-CabinBookingCounter, [354](#)
- AIRINV::InventoryParserHelper::storeSegment-CabinCode, [356](#)
- AIRINV::InventoryParserHelper::storeSegmentOff-Point, [359](#)
- AIRINV::InventoryParserHelper::storeSnapshot-Date, [363](#)
- AIRINV::InventoryParserHelper::storeSubclass-Code, [366](#)
- AIRINV::InventoryParserHelper::storeUPR, [367](#)
- AIRINV::InventoryParserHelper::storeYieldUpper-Range, [369](#)
- _flightDetails
 - AIRINV::Request, [227](#)
- _flightNumber
 - AIRINV::FlightDateStruct, [185](#)
 - AIRINV::FlightPeriodStruct, [194](#)
 - AIRINV::Request, [227](#)
 - stdair::BomPropertyTree, [133](#)
- _flightPeriod
 - AIRINV::ScheduleParserHelper::doEndFlight, [167](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-Parser, [190](#)
 - AIRINV::ScheduleParserHelper::ParserSemantic-Action, [222](#)
 - AIRINV::ScheduleParserHelper::storeAirlineCode, [252](#)
 - AIRINV::ScheduleParserHelper::storeBoarding-Time, [259](#)
 - AIRINV::ScheduleParserHelper::storeCapacity, [265](#)
 - AIRINV::ScheduleParserHelper::storeClasses, [273](#)
 - AIRINV::ScheduleParserHelper::storeDateRange-End, [279](#)
 - AIRINV::ScheduleParserHelper::storeDateRange-Start, [280](#)
 - AIRINV::ScheduleParserHelper::storeDow, [286](#)
 - AIRINV::ScheduleParserHelper::storeElapsed-Time, [287](#)
 - AIRINV::ScheduleParserHelper::storeFamilyCode, [293](#)
 - AIRINV::ScheduleParserHelper::storeFCClasses, [296](#)
 - AIRINV::ScheduleParserHelper::storeFlight-Number, [300](#)
 - AIRINV::ScheduleParserHelper::storeLegBoarding-Point, [308](#)
 - AIRINV::ScheduleParserHelper::storeLegCabin-Code, [309](#)
 - AIRINV::ScheduleParserHelper::storeLegOffPoint, [312](#)
 - AIRINV::ScheduleParserHelper::storeOffTime, [333](#)
 - AIRINV::ScheduleParserHelper::storeSegment-BoardingPoint, [353](#)
 - AIRINV::ScheduleParserHelper::storeSegment-CabinCode, [357](#)
 - AIRINV::ScheduleParserHelper::storeSegmentOff-Point, [360](#)
 - AIRINV::ScheduleParserHelper::storeSegment-Specificity, [362](#)
 - _flightTypeCode
 - AIRINV::FlightDateStruct, [186](#)
 - _flightVisibilityCode
 - AIRINV::FlightDateStruct, [186](#)
 - _gav
 - AIRINV::LegCabinStruct, [215](#)
 - _groupNbOfBookings
 - AIRINV::LegCabinStruct, [215](#)
 - _itBookingClass
 - AIRINV::FlightDateStruct, [188](#)
 - _itBucket
 - AIRINV::FlightDateStruct, [187](#)
 - _itCurrentAirlineCode
 - AIRINV::DCPEventStruct, [146](#)
 - _itCurrentClassCode
 - AIRINV::DCPEventStruct, [146](#)
 - _itDay
 - AIRINV::DCPEventStruct, [146](#)
 - AIRINV::FlightDateStruct, [186](#)
 - AIRINV::FlightPeriodStruct, [195](#)
 - _itFareFamily
 - AIRINV::SegmentCabinStruct, [234](#)
 - _itHours
 - AIRINV::DCPEventStruct, [146](#)
 - AIRINV::FlightDateStruct, [186](#)
 - AIRINV::FlightPeriodStruct, [195](#)
 - _itLeg
 - AIRINV::FlightDateStruct, [187](#)
 - AIRINV::FlightPeriodStruct, [194](#)
 - _itLegCabin
 - AIRINV::FlightDateStruct, [187](#)
 - AIRINV::FlightPeriodStruct, [194](#)
 - _itMinutes
 - AIRINV::DCPEventStruct, [146](#)
 - AIRINV::FlightDateStruct, [187](#)

- AIRINV::FlightPeriodStruct, 195
- _itMonth
 - AIRINV::DCPEventStruct, 146
 - AIRINV::FlightDateStruct, 186
 - AIRINV::FlightPeriodStruct, 195
- _itSeconds
 - AIRINV::DCPEventStruct, 146
 - AIRINV::FlightDateStruct, 187
 - AIRINV::FlightPeriodStruct, 195
- _itSegment
 - AIRINV::FlightDateStruct, 188
 - AIRINV::FlightPeriodStruct, 196
- _itSegmentCabin
 - AIRINV::FlightDateStruct, 188
 - AIRINV::FlightPeriodStruct, 196
- _itYear
 - AIRINV::DCPEventStruct, 146
 - AIRINV::FlightDateStruct, 186
 - AIRINV::FlightPeriodStruct, 195
- _legAlreadyDefined
 - AIRINV::FlightDateStruct, 187
 - AIRINV::FlightPeriodStruct, 194
- _legList
 - AIRINV::FlightDateStruct, 186
 - AIRINV::FlightPeriodStruct, 194
- _minimumStay
 - AIRINV::DCPEventStruct, 148
- _nav
 - AIRINV::LegCabinStruct, 215
- _nbOfBookings
 - AIRINV::BookingClassStruct, 136
 - AIRINV::LegCabinStruct, 215
 - AIRINV::SegmentCabinStruct, 234
- _nbOfFlights
 - AIRINV::InventoryParserHelper::doEndFlightDate, 169
 - AIRINV::InventoryParserHelper::InventoryParser, 212
- _nbOfGroupBookings
 - AIRINV::BookingClassStruct, 136
- _nbOfPendingGroupBookings
 - AIRINV::BookingClassStruct, 137
- _nbOfSeats
 - AIRINV::BucketStruct, 139
- _nbOfStaffBookings
 - AIRINV::BookingClassStruct, 137
- _nbOfWLBookings
 - AIRINV::BookingClassStruct, 137
- _nego
 - AIRINV::BookingClassStruct, 136
- _netClassAvailability
 - AIRINV::BookingClassStruct, 137
- _netRevenueAvailability
 - AIRINV::BookingClassStruct, 137
- _noShowPercentage
 - AIRINV::BookingClassStruct, 136
- _nonRefundable
 - AIRINV::DCPEventStruct, 148
- _offDate
 - AIRINV::LegStruct, 218
 - AIRINV::SegmentStruct, 238
- _offDateOffset
 - AIRINV::LegStruct, 218
- _offPoint
 - AIRINV::LegStruct, 218
 - AIRINV::SegmentStruct, 237
- _offTime
 - AIRINV::LegStruct, 218
 - AIRINV::SegmentStruct, 238
- _origin
 - AIRINV::DCPEventStruct, 147
- _overbookingPercentage
 - AIRINV::BookingClassStruct, 136
- _parentClassCode
 - AIRINV::BookingClassStruct, 136
- _parentSubclassCode
 - AIRINV::BookingClassStruct, 136
- _pool
 - AIRINV::FacBomAbstract, 175
 - AIRINV::FacServiceAbstract, 177
- _pos
 - AIRINV::DCPEventStruct, 147
- _protection
 - AIRINV::BookingClassStruct, 136
- _saleableCapacity
 - AIRINV::LegCabinStruct, 214
- _saturdayStay
 - AIRINV::DCPEventStruct, 148
- _seatIndex
 - AIRINV::BucketStruct, 139
- _segmentAvailability
 - AIRINV::BookingClassStruct, 137
- _segmentList
 - AIRINV::FlightDateStruct, 186
 - AIRINV::FlightPeriodStruct, 194
- _staffNbOfBookings
 - AIRINV::LegCabinStruct, 215
- _status
 - AIRINV::Reply, 226
- _subclassCode
 - AIRINV::BookingClassStruct, 136
- _timeRangeEnd
 - AIRINV::DCPEventStruct, 147
- _timeRangeStart
 - AIRINV::DCPEventStruct, 147
- _upr
 - AIRINV::LegCabinStruct, 215
- _wINbOfBookings
 - AIRINV::LegCabinStruct, 215
- _yieldRangeUpperValue
 - AIRINV::BucketStruct, 139
- AIRINV::FlightRequestStatus
 - INTERNAL_ERROR, 197
 - LAST_VALUE, 197
 - NOT_FOUND, 197
 - OK, 197

- AIRINV::FlightTypeCode
 - DOMESTIC, [199](#)
 - GROUND_HANDLING, [199](#)
 - INTERNATIONAL, [199](#)
 - LAST_VALUE, [199](#)
- AIRINV::FlightVisibilityCode
 - HIDDEN, [201](#)
 - LAST_VALUE, [201](#)
 - NORMAL, [201](#)
 - PSEUDO, [201](#)
- AIRINV, [103](#)
 - AIRINV_Master_ServicePtr_T, [106](#)
 - AIRINV_ServicePtr_Map_T, [106](#)
 - AIRINV_ServicePtr_T, [106](#)
 - AirportList_T, [107](#)
 - AirportOrderedList_T, [108](#)
 - BookingClassStructList_T, [108](#)
 - bounded1_2_p_t, [107](#)
 - bounded1_3_p_t, [107](#)
 - bounded1_4_p_t, [107](#)
 - bounded2_p_t, [107](#)
 - bounded4_p_t, [107](#)
 - BucketStructList_T, [108](#)
 - char_t, [106](#)
 - chset_t, [107](#)
 - ConnectionShrPtr_T, [109](#)
 - DepartureDateSegmentCabinMap_T, [108](#)
 - FRAT5Curve_T, [106](#)
 - FareFamilyStructList_T, [108](#)
 - int1_p_t, [106](#)
 - iterator_t, [106](#)
 - LegCabinStructList_T, [108](#)
 - LegStructList_T, [108](#)
 - repeat_p_t, [107](#)
 - rule_t, [106](#)
 - scanner_t, [106](#)
 - SegmentCabinStructList_T, [108](#)
 - SegmentStructList_T, [108](#)
 - SimilarSegmentCabinSetMap_T, [108](#)
 - ThreadShrPtr_T, [108](#)
 - ThreadShrPtrList_T, [108](#)
 - uint1_2_p_t, [107](#)
 - uint1_3_p_t, [107](#)
 - uint1_4_p_t, [107](#)
 - uint2_p_t, [106](#)
 - uint4_p_t, [107](#)
- AIRINV::AIRINV_Master_Service, [118](#)
 - buildSampleBom, [120](#)
 - calculateAvailability, [120](#)
 - cancel, [121](#)
 - check, [122](#)
 - csvDisplay, [122](#)
 - jsonExport, [121](#)
 - list, [121](#)
 - optimise, [121](#)
 - parseAndLoad, [119](#)
 - sell, [120](#)
 - takeSnapshots, [121](#)
- AIRINV::AIRINV_Master_ServiceContext, [123](#)
- AIRINV::AIRINV_Service, [123](#)
 - buildSampleBom, [126](#)
 - calculateAvailability, [126](#)
 - cancel, [126](#)
 - check, [128](#)
 - csvDisplay, [128](#)
 - initRMEvents, [126](#)
 - jsonExport, [127](#)
 - list, [127](#)
 - optimise, [127](#)
 - parseAndLoad, [125](#)
 - sell, [126](#)
 - takeSnapshots, [127](#)
- AIRINV::AIRINV_ServiceContext, [129](#)
 - FacAirinvServiceContext, [129](#)
- AIRINV::AirInvServer, [129](#)
 - ~AirInvServer, [130](#)
 - AirInvServer, [130](#)
 - run, [130](#)
 - stop, [130](#)
- AIRINV::BomAbstract, [130](#)
 - ~BomAbstract, [131](#)
 - BomAbstract, [131](#)
 - describeKey, [132](#)
 - describeShortKey, [132](#)
 - FacBomAbstract, [132](#)
 - fromStream, [131](#)
 - toStream, [131](#)
 - toString, [132](#)
- AIRINV::BomRootHelper, [133](#)
 - fillFromRouting, [134](#)
- AIRINV::BookingClassHelper, [134](#)
- AIRINV::BookingClassStruct, [134](#)
 - _classCode, [135](#)
 - _cumulatedProtection, [136](#)
 - _etb, [137](#)
 - _nbOfBookings, [136](#)
 - _nbOfGroupBookings, [136](#)
 - _nbOfPendingGroupBookings, [137](#)
 - _nbOfStaffBookings, [137](#)
 - _nbOfWLBookings, [137](#)
 - _nego, [136](#)
 - _netClassAvailability, [137](#)
 - _netRevenueAvailability, [137](#)
 - _noShowPercentage, [136](#)
 - _overbookingPercentage, [136](#)
 - _parentClassCode, [136](#)
 - _parentSubclassCode, [136](#)
 - _protection, [136](#)
 - _segmentAvailability, [137](#)
 - _subclassCode, [136](#)
 - BookingClassStruct, [135](#)
 - describe, [135](#)
 - fill, [135](#)
 - getFullSubclassCode, [135](#)
- AIRINV::BookingException, [137](#)
- AIRINV::BucketStruct, [138](#)

- [_availability](#), 139
 - [_nbOfSeats](#), 139
 - [_seatIndex](#), 139
 - [_yieldRangeUpperValue](#), 139
- [BucketStruct](#), 138
- [describe](#), 139
- [fill](#), 138
- [AIRINV::Connection](#), 141
 - [Connection](#), 141
 - [socket](#), 141
 - [start](#), 141
- [AIRINV::DCPEventGenerator](#), 142
 - [DCPFileParser](#), 142
 - [DCPPParser](#), 142
 - [DCPPParserHelper::doEndDCP](#), 142
- [AIRINV::DCPEventStruct](#), 143
 - [_DCP](#), 148
 - [_advancePurchase](#), 148
 - [_airlineCode](#), 148
 - [_airlineCodeList](#), 148
 - [_cabinCode](#), 147
 - [_changeFees](#), 148
 - [_channel](#), 147
 - [_classCode](#), 148
 - [_classCodeList](#), 149
 - [_dateRangeEnd](#), 147
 - [_dateRangeStart](#), 147
 - [_destination](#), 147
 - [_itCurrentAirlineCode](#), 146
 - [_itCurrentClassCode](#), 146
 - [_itDay](#), 146
 - [_itHours](#), 146
 - [_itMinutes](#), 146
 - [_itMonth](#), 146
 - [_itSeconds](#), 146
 - [_itYear](#), 146
 - [_minimumStay](#), 148
 - [_nonRefundable](#), 148
 - [_origin](#), 147
 - [_pos](#), 147
 - [_saturdayStay](#), 148
 - [_timeRangeEnd](#), 147
 - [_timeRangeStart](#), 147
- [beginAirline](#), 145
- [beginClassCode](#), 145
- [DCPEventStruct](#), 144
- [describe](#), 144
- [getAirlineListSize](#), 144
- [getClassCodeListSize](#), 144
- [getCurrentAirlineCode](#), 145
- [getCurrentClassCode](#), 145
- [getDate](#), 144
- [getFirstAirlineCode](#), 144
- [getFirstClassCode](#), 145
- [getTime](#), 144
- [hasNotReachedEndAirline](#), 145
- [hasNotReachedEndClassCode](#), 145
- [iterateAirline](#), 145
- [iterateClassCode](#), 146
- [AIRINV::DCPPParser](#), 149
 - [DCPRuleGeneration](#), 149
- [AIRINV::DCPPParserHelper](#), 109
 - [day_p](#), 111
 - [hour_p](#), 110
 - [int1_p](#), 110
 - [minute_p](#), 110
 - [month_p](#), 111
 - [second_p](#), 110
 - [uint1_4_p](#), 110
 - [uint2_p](#), 110
 - [uint4_p](#), 110
 - [year_p](#), 110
- [AIRINV::DCPPParserHelper::DCPRuleParser](#), 151
 - [_bomRoot](#), 156
 - [advancePurchase](#), 155
 - [cabinCode](#), 155
 - [changeFees](#), 155
 - [channel](#), 155
 - [comments](#), 153
 - [DCP](#), 155
 - [date](#), 154
 - [dateRangeEnd](#), 154
 - [dateRangeStart](#), 154
 - [destination](#), 154
 - [list_class](#), 156
 - [minimumStay](#), 155
 - [nonRefundable](#), 155
 - [origin](#), 154
 - [position](#), 154
 - [saturdayStay](#), 155
 - [segment](#), 155
 - [start](#), 153
 - [time](#), 154
 - [timeRangeEnd](#), 154
 - [timeRangeStart](#), 154
- [AIRINV::DCPPParserHelper::ParserSemanticAction](#), 223
 - [ParserSemanticAction](#), 224
- [AIRINV::DCPPParserHelper::doEndDCP](#), 165
 - [_bomRoot](#), 166
 - [doEndDCP](#), 166
 - [operator\(\)](#), 166
- [AIRINV::DCPPParserHelper::storeAdvancePurchase](#), 248
 - [_DCPRule](#), 249
 - [operator\(\)](#), 248
 - [storeAdvancePurchase](#), 248
- [AIRINV::DCPPParserHelper::storeAirlineCode](#), 252
 - [_DCPRule](#), 253
 - [operator\(\)](#), 253
 - [storeAirlineCode](#), 253
- [AIRINV::DCPPParserHelper::storeCabinCode](#), 262
 - [_DCPRule](#), 263
 - [operator\(\)](#), 263
 - [storeCabinCode](#), 263
- [AIRINV::DCPPParserHelper::storeChangeFees](#), 265
 - [_DCPRule](#), 266

- operator(), 266
- storeChangeFees, 266
- AIRINV::DCPPParserHelper::storeChannel, 266
 - _DCPRule, 267
 - operator(), 267
 - storeChannel, 267
- AIRINV::DCPPParserHelper::storeClass, 267
 - _DCPRule, 268
 - operator(), 268
 - storeClass, 268
- AIRINV::DCPPParserHelper::storeDCP, 282
 - operator(), 282
 - storeDCP, 282
- AIRINV::DCPPParserHelper::storeDCPId, 283
 - operator(), 283
 - storeDCPId, 283
- AIRINV::DCPPParserHelper::storeDateRangeEnd, 276
 - operator(), 277
 - storeDateRangeEnd, 277
- AIRINV::DCPPParserHelper::storeDateRangeStart, 280
 - operator(), 281
 - storeDateRangeStart, 281
- AIRINV::DCPPParserHelper::storeDestination, 284
 - _DCPRule, 285
 - operator(), 285
 - storeDestination, 285
- AIRINV::DCPPParserHelper::storeEndRangeTime, 288
 - operator(), 288
 - storeEndRangeTime, 288
- AIRINV::DCPPParserHelper::storeMinimumStay, 314
 - _DCPRule, 315
 - operator(), 315
 - storeMinimumStay, 315
- AIRINV::DCPPParserHelper::storeNonRefundable, 327
 - _DCPRule, 328
 - operator(), 327
 - storeNonRefundable, 327
- AIRINV::DCPPParserHelper::storeOrigin, 334
 - _DCPRule, 335
 - operator(), 335
 - storeOrigin, 334
- AIRINV::DCPPParserHelper::storePOS, 340
 - operator(), 341
 - storePOS, 340
- AIRINV::DCPPParserHelper::storeSaturdayStay, 346
 - _DCPRule, 347
 - operator(), 347
 - storeSaturdayStay, 346
- AIRINV::DCPPParserHelper::storeStartRangeTime, 364
 - operator(), 364
 - storeStartRangeTime, 364
- AIRINV::DCPRuleFileParser, 150
 - DCPRuleFileParser, 150
 - generateDCPRules, 150
- AIRINV::DefaultMap, 156
 - createPickupFRAT5Curve, 156
- AIRINV::FacAirinvMasterServiceContext, 170
 - ~FacAirinvMasterServiceContext, 171
- create, 171
- FacAirinvMasterServiceContext, 171
- instance, 171
- AIRINV::FacAirinvServiceContext, 172
 - ~FacAirinvServiceContext, 172
 - create, 173
 - FacAirinvServiceContext, 172
 - instance, 173
- AIRINV::FacBomAbstract, 173
 - ~FacBomAbstract, 174
 - _pool, 175
 - BomPool_T, 174
 - FacBomAbstract, 174
 - FacSupervisor, 175
 - getID, 174
 - getIDString, 174, 175
- AIRINV::FacServiceAbstract, 175
 - ~FacServiceAbstract, 176
 - _pool, 177
 - clean, 176
 - FacServiceAbstract, 176
 - ServicePool_T, 176
- AIRINV::FacSupervisor, 177
 - ~FacSupervisor, 178
 - BomFactoryPool_T, 177
 - cleanBomLayer, 179
 - cleanFactory, 179
 - cleanServiceLayer, 179
 - FacSupervisor, 178
 - instance, 178
 - registerBomFactory, 178
 - registerServiceFactory, 178
 - ServiceFactoryPool_T, 177
- AIRINV::FareFamilyStruct, 179
 - _classList, 180
 - _classes, 180
 - _familyCode, 180
 - describe, 180
 - FareFamilyStruct, 180
 - fill, 180
- AIRINV::FlightDateDuplicationException, 181
 - FlightDateDuplicationException, 181
- AIRINV::FlightDateHelper, 182
 - fillFromRouting, 182
 - updateAvailabilityPool, 182
 - updateBookingControls, 182
- AIRINV::FlightDateStruct, 183
 - _airlineCode, 185
 - _airportList, 187
 - _airportOrderedList, 187
 - _areSegmentDefinitionsSpecific, 188
 - _dateOffSet, 186
 - _flightDate, 186
 - _flightNumber, 185
 - _flightTypeCode, 186
 - _flightVisibilityCode, 186
 - _itBookingClass, 188
 - _itBucket, 187

- [_itDay, 186](#)
 - [_itHours, 186](#)
 - [_itLeg, 187](#)
 - [_itLegCabin, 187](#)
 - [_itMinutes, 187](#)
 - [_itMonth, 186](#)
 - [_itSeconds, 187](#)
 - [_itSegment, 188](#)
 - [_itSegmentCabin, 188](#)
 - [_itYear, 186](#)
 - [_legAlreadyDefined, 187](#)
 - [_legList, 186](#)
 - [_segmentList, 186](#)
- [addAirport, 184](#)
- [addFareFamily, 185](#)
- [addSegmentCabin, 185](#)
- [buildSegments, 184](#)
- [describe, 184](#)
- [FlightDateStruct, 184](#)
- [getDate, 184](#)
- [getTime, 184](#)
- [AIRINV::FlightPeriodFileParser, 188](#)
 - [FlightPeriodFileParser, 189](#)
- [AIRINV::FlightPeriodStruct, 191](#)
 - [_airlineCode, 194](#)
 - [_airportList, 195](#)
 - [_airportOrderedList, 196](#)
 - [_areSegmentDefinitionsSpecific, 196](#)
 - [_dateOffset, 195](#)
 - [_dateRange, 194](#)
 - [_dateRangeEnd, 195](#)
 - [_dateRangeStart, 195](#)
 - [_dow, 194](#)
 - [_flightNumber, 194](#)
 - [_itDay, 195](#)
 - [_itHours, 195](#)
 - [_itLeg, 194](#)
 - [_itLegCabin, 194](#)
 - [_itMinutes, 195](#)
 - [_itMonth, 195](#)
 - [_itSeconds, 195](#)
 - [_itSegment, 196](#)
 - [_itSegmentCabin, 196](#)
 - [_itYear, 195](#)
 - [_legAlreadyDefined, 194](#)
 - [_legList, 194](#)
 - [_segmentList, 194](#)
- [addAirport, 192](#)
- [addFareFamily, 193](#)
- [addSegmentCabin, 193](#)
- [buildSegments, 193](#)
- [describe, 192](#)
- [FlightPeriodStruct, 192](#)
- [getDate, 192](#)
- [getTime, 192](#)
- [AIRINV::FlightRequestStatus, 196](#)
 - [describe, 198](#)
 - [describeLabels, 198](#)
 - [EN_FlightRequestStatus, 197](#)
 - [FlightRequestStatus, 197](#)
 - [getCode, 198](#)
 - [getCodeLabel, 197](#)
 - [getLabel, 197](#)
- [AIRINV::FlightTypeCode, 198](#)
 - [describe, 200](#)
 - [describeLabels, 199](#)
 - [EN_FlightTypeCode, 199](#)
 - [FlightTypeCode, 199](#)
 - [getCode, 200](#)
 - [getCodeLabel, 199](#)
 - [getLabel, 199](#)
- [AIRINV::FlightVisibilityCode, 200](#)
 - [describe, 201](#)
 - [describeLabels, 201](#)
 - [EN_FlightVisibilityCode, 201](#)
 - [FlightVisibilityCode, 201](#)
 - [getCode, 201](#)
 - [getCodeLabel, 201](#)
 - [getLabel, 201](#)
- [AIRINV::GuillotineBlockHelper, 202](#)
 - [takeSnapshots, 203](#)
- [AIRINV::InventoryBuilder, 203](#)
 - [InventoryParserHelper::doEndFlightDate, 204](#)
- [AIRINV::InventoryFileParser, 204](#)
 - [buildInventory, 205](#)
 - [InventoryFileParser, 204](#)
- [AIRINV::InventoryFileParsingFailedException, 205](#)
 - [InventoryFileParsingFailedException, 205](#)
- [AIRINV::InventoryGenerator, 205](#)
 - [FFFlightPeriodFileParser, 206](#)
 - [FlightPeriodFileParser, 206](#)
 - [ScheduleParser, 206](#)
 - [ScheduleParserHelper::doEndFlight, 206](#)
- [AIRINV::InventoryHelper, 206](#)
 - [calculateAvailability, 207](#)
 - [cancel, 207](#)
 - [fillFromRouting, 207](#)
 - [getYieldAndBidPrice, 207](#)
 - [sell, 207](#)
 - [takeSnapshots, 207](#)
- [AIRINV::InventoryInputFileNotFoundException, 208](#)
 - [InventoryInputFileNotFoundException, 208](#)
- [AIRINV::InventoryManager, 208](#)
 - [AIRINV_Master_Service, 210](#)
 - [AIRINV_Service, 210](#)
 - [buildGuillotineBlock, 209](#)
 - [buildSimilarSegmentCabinSets, 209](#)
 - [createDirectAccesses, 209](#)
 - [setDefaultBidPriceVector, 209, 210](#)
- [AIRINV::InventoryParser, 210](#)
 - [buildInventory, 210](#)
- [AIRINV::InventoryParserHelper, 111](#)
 - [airline_code_p, 112](#)
 - [airport_p, 113](#)
 - [cabin_code_p, 113](#)

- class_code_list_p, [113](#)
- class_code_p, [113](#)
- day_p, [113](#)
- dow_p, [113](#)
- family_code_p, [114](#)
- flight_number_p, [112](#)
- hours_p, [113](#)
- int1_p, [114](#)
- minutes_p, [113](#)
- month_p, [113](#)
- passenger_type_p, [113](#)
- seconds_p, [113](#)
- stay_duration_p, [113](#)
- uint1_2_p, [114](#)
- uint1_3_p, [114](#)
- uint1_4_p, [114](#)
- uint2_p, [114](#)
- uint4_p, [114](#)
- year_p, [112](#)
- AIRINV::InventoryParserHelper::InventoryParser, [211](#)
 - _bomRoot, [212](#)
 - _flightDate, [212](#)
 - _nbOfFlights, [212](#)
 - InventoryParser, [211](#)
- AIRINV::InventoryParserHelper::InventoryParser-
::definition
 - airline_code, [159](#)
 - bucket_details, [160](#)
 - bucket_list, [160](#)
 - class_details, [161](#)
 - class_key, [161](#)
 - class_list, [161](#)
 - class_nego, [161](#)
 - class_protection, [161](#)
 - date, [159](#)
 - definition, [158](#)
 - family_cabin_details, [161](#)
 - family_cabin_list, [161](#)
 - flight_date, [159](#)
 - flight_date_end, [159](#)
 - flight_date_list, [159](#)
 - flight_key, [159](#)
 - flight_number, [159](#)
 - flight_type_code, [159](#)
 - flight_visibility_code, [159](#)
 - leg, [159](#)
 - leg_cabin_details, [160](#)
 - leg_cabin_list, [160](#)
 - leg_details, [160](#)
 - leg_key, [159](#)
 - leg_list, [159](#)
 - not_to_be_parsed, [159](#)
 - parent_subclass_code, [161](#)
 - segment, [160](#)
 - segment_cabin_details, [161](#)
 - segment_cabin_key, [160](#)
 - segment_cabin_list, [160](#)
 - segment_key, [160](#)
 - segment_list, [160](#)
 - start, [158](#)
 - time, [160](#)
- AIRINV::InventoryParserHelper::InventoryParser-
::definition< ScannerT >, [157](#)
- AIRINV::InventoryParserHelper::ParserSemanticAction, [219](#)
 - _flightDate, [220](#)
 - ParserSemanticAction, [220](#)
- AIRINV::InventoryParserHelper::doEndFlightDate, [168](#)
 - _bomRoot, [169](#)
 - _flightDate, [169](#)
 - _nbOfFlights, [169](#)
 - doEndFlightDate, [169](#)
 - operator(), [169](#)
- AIRINV::InventoryParserHelper::storeACP, [246](#)
 - _flightDate, [247](#)
 - operator(), [247](#)
 - storeACP, [247](#)
- AIRINV::InventoryParserHelper::storeAU, [253](#)
 - _flightDate, [254](#)
 - operator(), [254](#)
 - storeAU, [254](#)
- AIRINV::InventoryParserHelper::storeAirlineCode, [249](#)
 - _flightDate, [250](#)
 - operator(), [250](#)
 - storeAirlineCode, [250](#)
- AIRINV::InventoryParserHelper::storeBoardingDate, [255](#)
 - _flightDate, [256](#)
 - operator(), [255](#)
 - storeBoardingDate, [255](#)
- AIRINV::InventoryParserHelper::storeBoardingTime, [256](#)
 - _flightDate, [257](#)
 - operator(), [257](#)
 - storeBoardingTime, [257](#)
- AIRINV::InventoryParserHelper::storeBookingCounter, [259](#)
 - _flightDate, [260](#)
 - operator(), [260](#)
 - storeBookingCounter, [260](#)
- AIRINV::InventoryParserHelper::storeBucketAvaibility, [261](#)
 - _flightDate, [262](#)
 - operator(), [262](#)
 - storeBucketAvaibility, [261](#)
- AIRINV::InventoryParserHelper::storeClassAvailability, [269](#)
 - _flightDate, [270](#)
 - operator(), [269](#)
 - storeClassAvailability, [269](#)
- AIRINV::InventoryParserHelper::storeClassCode, [270](#)
 - _flightDate, [271](#)
 - operator(), [271](#)
 - storeClassCode, [271](#)
- AIRINV::InventoryParserHelper::storeClassETB, [273](#)
 - _flightDate, [274](#)

- operator(), 274
- storeClassETB, 274
- AIRINV::InventoryParserHelper::storeCumulated-
Protection, 275
 - _flightDate, 276
- operator(), 275
- storeCumulatedProtection, 275
- AIRINV::InventoryParserHelper::storeETB, 289
 - _flightDate, 290
- operator(), 290
- storeETB, 290
- AIRINV::InventoryParserHelper::storeFCClasses, 294
 - _flightDate, 294
- operator(), 294
- storeFCClasses, 294
- AIRINV::InventoryParserHelper::storeFamilyCode, 291
 - _flightDate, 292
- operator(), 291
- storeFamilyCode, 291
- AIRINV::InventoryParserHelper::storeFlightDate, 296
 - _flightDate, 297
- operator(), 297
- storeFlightDate, 297
- AIRINV::InventoryParserHelper::storeFlightNumber, 298
 - _flightDate, 299
- operator(), 299
- storeFlightNumber, 299
- AIRINV::InventoryParserHelper::storeFlightTypeCode, 301
 - _flightDate, 302
- operator(), 302
- storeFlightTypeCode, 301
- AIRINV::InventoryParserHelper::storeFlightVisibility-
Code, 302
 - _flightDate, 303
- operator(), 303
- storeFlightVisibilityCode, 303
- AIRINV::InventoryParserHelper::storeGAV, 304
 - _flightDate, 305
- operator(), 305
- storeGAV, 305
- AIRINV::InventoryParserHelper::storeLegBoarding-
Point, 306
 - _flightDate, 307
- operator(), 306
- storeLegBoardingPoint, 306
- AIRINV::InventoryParserHelper::storeLegCabinCode, 310
 - _flightDate, 311
- operator(), 311
- storeLegCabinCode, 310
- AIRINV::InventoryParserHelper::storeLegOffPoint, 313
 - _flightDate, 314
- operator(), 313
- storeLegOffPoint, 313
- AIRINV::InventoryParserHelper::storeNAV, 316
 - _flightDate, 317
- operator(), 316
- storeNAV, 316
- AIRINV::InventoryParserHelper::storeNbOfBkgs, 317
 - _flightDate, 318
- operator(), 318
- storeNbOfBkgs, 318
- AIRINV::InventoryParserHelper::storeNbOfGroupBkgs, 319
 - _flightDate, 320
- operator(), 320
- storeNbOfGroupBkgs, 319
- AIRINV::InventoryParserHelper::storeNbOfPending-
GroupBkgs, 320
 - _flightDate, 321
- operator(), 321
- AIRINV::InventoryParserHelper::storeNbOfStaffBkgs, 322
 - _flightDate, 323
- operator(), 323
- storeNbOfStaffBkgs, 323
- AIRINV::InventoryParserHelper::storeNbOfWLBkgs, 324
 - _flightDate, 324
- operator(), 324
- storeNbOfWLBkgs, 324
- AIRINV::InventoryParserHelper::storeNego, 325
 - _flightDate, 326
- operator(), 326
- storeNego, 326
- AIRINV::InventoryParserHelper::storeNoShow, 328
 - _flightDate, 329
- operator(), 329
- storeNoShow, 328
- AIRINV::InventoryParserHelper::storeOffDate, 329
 - _flightDate, 330
- operator(), 330
- storeOffDate, 330
- AIRINV::InventoryParserHelper::storeOffTime, 331
 - _flightDate, 332
- operator(), 332
- storeOffTime, 332
- AIRINV::InventoryParserHelper::storeOverbooking, 335
 - _flightDate, 336
- operator(), 336
- storeOverbooking, 336
- AIRINV::InventoryParserHelper::storeParentClassCode, 337
 - _flightDate, 338
- operator(), 337
- storeParentClassCode, 337
- AIRINV::InventoryParserHelper::storeParentSubclass-
Code, 338
 - _flightDate, 339
- operator(), 339
- storeParentSubclassCode, 339
- AIRINV::InventoryParserHelper::storeProtection, 341
 - _flightDate, 342
- operator(), 342

- storeProtection, 342
- AIRINV::InventoryParserHelper::storeRevenueAvailability, 343
 - _flightDate, 344
 - operator(), 343
 - storeRevenueAvailability, 343
- AIRINV::InventoryParserHelper::storeSaleableCapacity, 344
 - _flightDate, 345
 - operator(), 345
 - storeSaleableCapacity, 345
- AIRINV::InventoryParserHelper::storeSeatIndex, 347
 - _flightDate, 348
 - operator(), 348
 - storeSeatIndex, 348
- AIRINV::InventoryParserHelper::storeSegmentAvailability, 349
 - _flightDate, 350
 - operator(), 349
 - storeSegmentAvailability, 349
- AIRINV::InventoryParserHelper::storeSegmentBoarding-Point, 350
 - _flightDate, 351
 - operator(), 351
 - storeSegmentBoardingPoint, 351
- AIRINV::InventoryParserHelper::storeSegmentCabin-BookingCounter, 353
 - _flightDate, 354
 - operator(), 354
 - storeSegmentCabinBookingCounter, 354
- AIRINV::InventoryParserHelper::storeSegmentCabin-Code, 355
 - _flightDate, 356
 - operator(), 356
 - storeSegmentCabinCode, 355
- AIRINV::InventoryParserHelper::storeSegmentOffPoint, 358
 - _flightDate, 359
 - operator(), 359
 - storeSegmentOffPoint, 358
- AIRINV::InventoryParserHelper::storeSnapshotDate, 362
 - _flightDate, 363
 - operator(), 363
 - storeSnapshotDate, 363
- AIRINV::InventoryParserHelper::storeSubclassCode, 365
 - _flightDate, 366
 - operator(), 366
 - storeSubclassCode, 365
- AIRINV::InventoryParserHelper::storeUPR, 366
 - _flightDate, 367
 - operator(), 367
 - storeUPR, 367
- AIRINV::InventoryParserHelper::storeYieldUpper-Range, 368
 - _flightDate, 369
 - operator(), 369
 - storeYieldUpperRange, 369
- AIRINV::LegCabinHelper, 213
- AIRINV::LegCabinStruct, 213
 - _acp, 215
 - _adjustment, 214
 - _au, 215
 - _avPool, 215
 - _bucketList, 216
 - _cabinCode, 214
 - _dcsRegrade, 214
 - _etb, 215
 - _gav, 215
 - _groupNbOfBookings, 215
 - _nav, 215
 - _nbOfBookings, 215
 - _saleableCapacity, 214
 - _staffNbOfBookings, 215
 - _upr, 215
 - _wlnNbOfBookings, 215
 - describe, 214
 - fill, 214
- AIRINV::LegStruct, 216
 - _boardingDate, 217
 - _boardingDateOffset, 217
 - _boardingPoint, 217
 - _boardingTime, 217
 - _cabinList, 218
 - _elapsed, 218
 - _offDate, 218
 - _offDateOffset, 218
 - _offPoint, 218
 - _offTime, 218
 - describe, 217
 - fill, 217
 - LegStruct, 217
- AIRINV::Reply, 225
 - _status, 226
 - content, 226
 - to_buffers, 226
- AIRINV::Request, 226
 - _airlineCode, 227
 - _departureDate, 227
 - _flightDetails, 227
 - _flightNumber, 227
 - parseFlightDate, 227
- AIRINV::RequestHandler, 227
 - handleRequest, 228
 - RequestHandler, 228
- AIRINV::RequestParser, 228
 - parse, 229
 - RequestParser, 229
 - reset, 229
- AIRINV::ScheduleFileParsingFailedException, 230
 - ScheduleFileParsingFailedException, 230
- AIRINV::ScheduleInputFileNotFoundException, 230
 - ScheduleInputFileNotFoundException, 231
- AIRINV::ScheduleParser, 231
 - generateInventories, 231

- AIRINV::ScheduleParserHelper, [114](#)
 - airline_code_p, [115](#)
 - airport_p, [116](#)
 - cabin_code_p, [116](#)
 - class_code_list_p, [116](#)
 - day_p, [116](#)
 - dow_p, [116](#)
 - family_code_p, [117](#)
 - flight_number_p, [115](#)
 - hours_p, [116](#)
 - int1_p, [116](#)
 - minutes_p, [116](#)
 - month_p, [116](#)
 - seconds_p, [116](#)
 - uint1_4_p, [117](#)
 - uint2_p, [117](#)
 - uint4_p, [117](#)
 - year_p, [115](#)
- AIRINV::ScheduleParserHelper::FlightPeriodParser, [189](#)
 - _bomRoot, [190](#)
 - _flightPeriod, [190](#)
 - FlightPeriodParser, [190](#)
- AIRINV::ScheduleParserHelper::FlightPeriodParser-
::definition
 - airline_code, [163](#)
 - date, [164](#)
 - date_offset, [164](#)
 - definition, [163](#)
 - dow, [164](#)
 - family_cabin_details, [165](#)
 - flight_key, [163](#)
 - flight_number, [164](#)
 - flight_period, [163](#)
 - flight_period_end, [163](#)
 - flight_period_list, [163](#)
 - generic_segment, [165](#)
 - leg, [164](#)
 - leg_cabin_details, [164](#)
 - leg_details, [164](#)
 - leg_key, [164](#)
 - segment_cabin_details, [165](#)
 - segment_key, [164](#)
 - segment_section, [164](#)
 - specific_segment_list, [165](#)
 - start, [163](#)
 - time, [164](#)
- AIRINV::ScheduleParserHelper::FlightPeriodParser-
::definition< ScannerT >, [161](#)
- AIRINV::ScheduleParserHelper::ParserSemanticAction, [221](#)
 - _flightPeriod, [222](#)
 - ParserSemanticAction, [222](#)
- AIRINV::ScheduleParserHelper::doEndFlight, [166](#)
 - _bomRoot, [167](#)
 - _flightPeriod, [167](#)
 - doEndFlight, [167](#)
 - operator(), [167](#)
- AIRINV::ScheduleParserHelper::storeAirlineCode, [251](#)
 - _flightPeriod, [252](#)
 - operator(), [251](#)
 - storeAirlineCode, [251](#)
- AIRINV::ScheduleParserHelper::storeBoardingTime, [258](#)
 - _flightPeriod, [259](#)
 - operator(), [259](#)
 - storeBoardingTime, [259](#)
- AIRINV::ScheduleParserHelper::storeCapacity, [264](#)
 - _flightPeriod, [265](#)
 - operator(), [264](#)
 - storeCapacity, [264](#)
- AIRINV::ScheduleParserHelper::storeClasses, [272](#)
 - _flightPeriod, [273](#)
 - operator(), [273](#)
 - storeClasses, [272](#)
- AIRINV::ScheduleParserHelper::storeDateRangeEnd, [278](#)
 - _flightPeriod, [279](#)
 - operator(), [278](#)
 - storeDateRangeEnd, [278](#)
- AIRINV::ScheduleParserHelper::storeDateRangeStart, [279](#)
 - _flightPeriod, [280](#)
 - operator(), [280](#)
 - storeDateRangeStart, [280](#)
- AIRINV::ScheduleParserHelper::storeDow, [285](#)
 - _flightPeriod, [286](#)
 - operator(), [286](#)
 - storeDow, [286](#)
- AIRINV::ScheduleParserHelper::storeElapsedTime, [286](#)
 - _flightPeriod, [287](#)
 - operator(), [287](#)
 - storeElapsedTime, [287](#)
- AIRINV::ScheduleParserHelper::storeFCClasses, [295](#)
 - _flightPeriod, [296](#)
 - operator(), [296](#)
 - storeFCClasses, [296](#)
- AIRINV::ScheduleParserHelper::storeFamilyCode, [292](#)
 - _flightPeriod, [293](#)
 - operator(), [293](#)
 - storeFamilyCode, [293](#)
- AIRINV::ScheduleParserHelper::storeFlightNumber, [300](#)
 - _flightPeriod, [300](#)
 - operator(), [300](#)
 - storeFlightNumber, [300](#)
- AIRINV::ScheduleParserHelper::storeLegBoarding-
Point, [307](#)
 - _flightPeriod, [308](#)
 - operator(), [308](#)
 - storeLegBoardingPoint, [308](#)
- AIRINV::ScheduleParserHelper::storeLegCabinCode, [309](#)
 - _flightPeriod, [309](#)
 - operator(), [309](#)
 - storeLegCabinCode, [309](#)

- AIRINV::ScheduleParserHelper::storeLegOffPoint, 311
 - _flightPeriod, 312
 - operator(), 312
 - storeLegOffPoint, 312
- AIRINV::ScheduleParserHelper::storeOffTime, 333
 - _flightPeriod, 333
 - operator(), 333
 - storeOffTime, 333
- AIRINV::ScheduleParserHelper::storeSegmentBoarding-Point, 352
 - _flightPeriod, 353
 - operator(), 353
 - storeSegmentBoardingPoint, 352
- AIRINV::ScheduleParserHelper::storeSegmentCabin-Code, 357
 - _flightPeriod, 357
 - operator(), 357
 - storeSegmentCabinCode, 357
- AIRINV::ScheduleParserHelper::storeSegmentOffPoint, 359
 - _flightPeriod, 360
 - operator(), 360
 - storeSegmentOffPoint, 360
- AIRINV::ScheduleParserHelper::storeSegmentSpecificity, 361
 - _flightPeriod, 362
 - operator(), 361
 - storeSegmentSpecificity, 361
- AIRINV::SegmentCabinHelper, 232
 - buildPseudoBidPriceVector, 232
 - initialiseAU, 233
 - updateAUs, 232
 - updateAvailabilities, 233
 - updateBookingControlsUsingPseudoBidPrice-Vector, 232
 - updateFromReservation, 232
- AIRINV::SegmentCabinStruct, 233
 - _cabinCode, 234
 - _fareFamilies, 234
 - _itFareFamily, 234
 - _nbOfBookings, 234
 - describe, 234
 - fill, 234
- AIRINV::SegmentDateHelper, 235
 - fillFromRouting, 235
 - updateDistanceFromElapsedTime, 235
 - updateElapsedTimeFromRouting, 235
- AIRINV::SegmentDateNotFoundException, 236
 - SegmentDateNotFoundException, 236
- AIRINV::SegmentStruct, 236
 - _boardingDate, 237
 - _boardingPoint, 237
 - _boardingTime, 238
 - _cabinList, 238
 - _elapsed, 238
 - _offDate, 238
 - _offPoint, 237
 - _offTime, 238
 - describe, 237
 - fill, 237
- AIRINV::ServiceAbstract, 238
 - ~ServiceAbstract, 239
 - fromStream, 239
 - ServiceAbstract, 239
 - toStream, 239
- AIRINV::header, 203
 - name, 203
 - value, 203
- AIRINV_Master_Service
 - AIRINV::AIRINV_Master_Service, 118, 119
 - AIRINV::InventoryManager, 210
- AIRINV_Master_ServicePtr_T
 - AIRINV, 106
- AIRINV_Service
 - AIRINV::AIRINV_Service, 124, 125
 - AIRINV::AIRINV_ServiceContext, 129
 - AIRINV::InventoryManager, 210
- AIRINV_ServicePtr_Map_T
 - AIRINV, 106
- AIRINV_ServicePtr_T
 - AIRINV, 106
- addAirport
 - AIRINV::FlightDateStruct, 184
 - AIRINV::FlightPeriodStruct, 192
- addFareFamily
 - AIRINV::FlightDateStruct, 185
 - AIRINV::FlightPeriodStruct, 193
- addSegmentCabin
 - AIRINV::FlightDateStruct, 185
 - AIRINV::FlightPeriodStruct, 193
- advancePurchase
 - AIRINV::DCPParserHelper::DCPRuleParser, 155
- AirInvClient.cpp
 - main, 537
- AirInvClient_ASIO.cpp
 - main, 538
- AirInvServer
 - AIRINV::AirInvServer, 130
- airinv-paths.hpp
 - BINDIR, 524
 - DATADIR, 524
 - DATAROOTDIR, 524
 - DOCDIR, 524
 - EXEC_PREFIX, 524
 - HTMLDIR, 525
 - INCLUDEDIR, 524
 - INFODIR, 524
 - LIBDIR, 524
 - LIBEXECDIR, 524
 - MANDIR, 524
 - PACKAGE, 524
 - PACKAGE_NAME, 524
 - PACKAGE_VERSION, 524
 - PDFDIR, 525
 - PREFIXDIR, 524
 - SBINDIR, 524

- STDAIR_SAMPLE_DIR, 525
- SYSCONFDIR, 524
- airinv-paths.hpp.in
 - BINDIR, 526
 - DATADIR, 526
 - DATAROOTDIR, 526
 - DOCDIR, 526
 - EXEC_PREFIX, 526
 - HTMLDIR, 527
 - INCLUDEDIR, 526
 - INFODIR, 527
 - LIBDIR, 526
 - LIBEXECDIR, 526
 - MANDIR, 527
 - PACKAGE, 526
 - PACKAGE_NAME, 526
 - PACKAGE_VERSION, 526
 - PDFDIR, 527
 - PREFIXDIR, 526
 - SBINDIR, 526
 - STDAIR_SAMPLE_DIR, 527
 - SYSCONFDIR, 526
- airinv/ Directory Reference, 99
- airinv/AIRINV_Master_Service.hpp, 371
- airinv/AIRINV_Service.hpp, 373
- airinv/AIRINV_Types.hpp, 375
- airinv/FlightRequestStatus.hpp, 536
- airinv/basic/ Directory Reference, 99
- airinv/basic/BasConst.cpp, 376
- airinv/basic/BasConst_AIRINV_Service.hpp, 378
- airinv/basic/BasConst_Curves.hpp, 378
- airinv/basic/BasConst_General.hpp, 379
- airinv/basic/BasParserTypes.hpp, 379, 380
- airinv/basic/FlightRequestStatus.cpp, 381
- airinv/basic/FlightTypeCode.cpp, 383
- airinv/basic/FlightTypeCode.hpp, 384
- airinv/basic/FlightVisibilityCode.cpp, 385
- airinv/basic/FlightVisibilityCode.hpp, 386, 387
- airinv/batches/ Directory Reference, 99
- airinv/batches/airinv_parseInventory.cpp, 387
- airinv/batches/parseInventory.cpp, 391
- airinv/bom/ Directory Reference, 99
- airinv/bom/AirportList.hpp, 395
- airinv/bom/BomAbstract.cpp, 396
- airinv/bom/BomAbstract.hpp, 396, 397
- airinv/bom/BomRootHelper.cpp, 398
- airinv/bom/BomRootHelper.hpp, 398
- airinv/bom/BookingClassHelper.cpp, 399
- airinv/bom/BookingClassHelper.hpp, 399
- airinv/bom/BookingClassStruct.cpp, 400
- airinv/bom/BookingClassStruct.hpp, 401
- airinv/bom/BucketStruct.cpp, 402
- airinv/bom/BucketStruct.hpp, 403
- airinv/bom/DCPEEventStruct.cpp, 404
- airinv/bom/DCPEEventStruct.hpp, 406
- airinv/bom/FareFamilyStruct.cpp, 408
- airinv/bom/FareFamilyStruct.hpp, 409
- airinv/bom/FlightDateHelper.cpp, 410
- airinv/bom/FlightDateHelper.hpp, 411
- airinv/bom/FlightDateStruct.cpp, 412
- airinv/bom/FlightDateStruct.hpp, 415, 416
- airinv/bom/FlightPeriodStruct.cpp, 417
- airinv/bom/FlightPeriodStruct.hpp, 420, 421
- airinv/bom/GuillotineBlockHelper.cpp, 422
- airinv/bom/GuillotineBlockHelper.hpp, 425
- airinv/bom/InventoryHelper.cpp, 426
- airinv/bom/InventoryHelper.hpp, 431
- airinv/bom/LegCabinHelper.cpp, 432
- airinv/bom/LegCabinHelper.hpp, 432
- airinv/bom/LegCabinStruct.cpp, 433
- airinv/bom/LegCabinStruct.hpp, 433, 434
- airinv/bom/LegStruct.cpp, 434, 435
- airinv/bom/LegStruct.hpp, 436
- airinv/bom/SegmentCabinHelper.cpp, 437
- airinv/bom/SegmentCabinHelper.hpp, 439, 440
- airinv/bom/SegmentCabinStruct.cpp, 440, 441
- airinv/bom/SegmentCabinStruct.hpp, 441, 442
- airinv/bom/SegmentDateHelper.cpp, 442
- airinv/bom/SegmentDateHelper.hpp, 444
- airinv/bom/SegmentStruct.cpp, 444, 445
- airinv/bom/SegmentStruct.hpp, 445, 446
- airinv/command/ Directory Reference, 100
- airinv/command/InventoryBuilder.cpp, 446, 447
- airinv/command/InventoryBuilder.hpp, 451
- airinv/command/InventoryGenerator.cpp, 452, 453
- airinv/command/InventoryGenerator.hpp, 457
- airinv/command/InventoryManager.cpp, 458, 459
- airinv/command/InventoryManager.hpp, 472, 473
- airinv/command/InventoryParser.cpp, 474
- airinv/command/InventoryParser.hpp, 475
- airinv/command/InventoryParserHelper.cpp, 475, 476
- airinv/command/InventoryParserHelper.hpp, 489, 491
- airinv/command/ScheduleParser.cpp, 495
- airinv/command/ScheduleParser.hpp, 496
- airinv/command/ScheduleParserHelper.cpp, 497
- airinv/command/ScheduleParserHelper.hpp, 505, 506
- airinv/command/vault/ Directory Reference, 102
- airinv/command/vault/DCPEEventGenerator.cpp, 508, 509
- airinv/command/vault/DCPEEventGenerator.hpp, 510
- airinv/command/vault/DCPPParser.cpp, 510, 511
- airinv/command/vault/DCPPParser.hpp, 511
- airinv/command/vault/DCPPParserHelper.cpp, 512
- airinv/command/vault/DCPPParserHelper.hpp, 520, 521
- airinv/config/ Directory Reference, 101
- airinv/config/airinv-paths.hpp, 523, 525
- airinv/config/airinv-paths.hpp.in, 525, 527
- airinv/factory/ Directory Reference, 101
- airinv/factory/FacAirinvMasterServiceContext.cpp, 527, 528
- airinv/factory/FacAirinvMasterServiceContext.hpp, 528
- airinv/factory/FacAirinvServiceContext.cpp, 529
- airinv/factory/FacAirinvServiceContext.hpp, 530
- airinv/factory/FacBomAbstract.cpp, 531
- airinv/factory/FacBomAbstract.hpp, 532
- airinv/factory/FacServiceAbstract.cpp, 532, 533

- airinv/factory/FacServiceAbstract.hpp, 533
- airinv/factory/FacSupervisor.cpp, 534
- airinv/factory/FacSupervisor.hpp, 535
- airinv/server/ Directory Reference, 101
- airinv/server/AirInvClient.cpp, 537
- airinv/server/AirInvClient_ASIO.cpp, 538
- airinv/server/AirInvServer.cpp, 539
- airinv/server/AirInvServer.hpp, 544
- airinv/server/AirInvServer_ASIO.cpp, 545
- airinv/server/BomPropertyTree.cpp, 546, 547
- airinv/server/BomPropertyTree.hpp, 548
- airinv/server/Connection.cpp, 549
- airinv/server/Connection.hpp, 550, 551
- airinv/server/Reply.cpp, 553
- airinv/server/Reply.hpp, 554
- airinv/server/Request.cpp, 554, 555
- airinv/server/Request.hpp, 555
- airinv/server/RequestHandler.cpp, 556
- airinv/server/RequestHandler.hpp, 557
- airinv/server/RequestParser.cpp, 558
- airinv/server/RequestParser.hpp, 561
- airinv/server/header.hpp, 551, 552
- airinv/server/posix_main.cpp, 552
- airinv/server/win_main.cpp, 562
- airinv/service/ Directory Reference, 102
- airinv/service/AIRINV_Master_Service.cpp, 563, 564
- airinv/service/AIRINV_Master_ServiceContext.cpp, 571
- airinv/service/AIRINV_Master_ServiceContext.hpp, 572
- airinv/service/AIRINV_Service.cpp, 573, 574
- airinv/service/AIRINV_ServiceContext.cpp, 582
- airinv/service/AIRINV_ServiceContext.hpp, 583
- airinv/service/ServiceAbstract.cpp, 584
- airinv/service/ServiceAbstract.hpp, 585
- airinv/ui/ Directory Reference, 102
- airinv/ui/cmdline/ Directory Reference, 100
- airinv/ui/cmdline/SReadline.hpp, 606
- airinv/ui/cmdline/airinv.cpp, 586
- airinv/ui/cmdline/readline_autocomp.hpp, 598, 602
- airline_code
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 159
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 163
- airline_code_p
 - AIRINV::InventoryParserHelper, 112
 - AIRINV::ScheduleParserHelper, 115
- airport_p
 - AIRINV::InventoryParserHelper, 113
 - AIRINV::ScheduleParserHelper, 116
- AirportList_T
 - AIRINV, 107
- AirportOrderedList_T
 - AIRINV, 108
- BINDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- beginAirline
 - AIRINV::DCPEventStruct, 145
- beginClassCode
 - AIRINV::DCPEventStruct, 145
- Bind
 - swift::SKeymap, 241
- BomAbstract
 - AIRINV::BomAbstract, 131
- BomAbstract.hpp
 - operator<<, 396
 - operator>>, 397
- BomFactoryPool_T
 - AIRINV::FacSupervisor, 177
- BomPool_T
 - AIRINV::FacBomAbstract, 174
- BookingClassStruct
 - AIRINV::BookingClassStruct, 135
- BookingClassStructList_T
 - AIRINV, 108
- bounded1_2_p_t
 - AIRINV, 107
- bounded1_3_p_t
 - AIRINV, 107
- bounded1_4_p_t
 - AIRINV, 107
- bounded2_p_t
 - AIRINV, 107
- bounded4_p_t
 - AIRINV, 107
- bucket_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 160
- bucket_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 160
- BucketStruct
 - AIRINV::BucketStruct, 138
- BucketStructList_T
 - AIRINV, 108
- buildGuillotineBlock
 - AIRINV::InventoryManager, 209
- buildInventory
 - AIRINV::InventoryFileParser, 205
 - AIRINV::InventoryParser, 210
- buildPseudoBidPriceVector
 - AIRINV::SegmentCabinHelper, 232
- buildSampleBom
 - AIRINV::AIRINV_Master_Service, 120
 - AIRINV::AIRINV_Service, 126
- buildSegments
 - AIRINV::FlightDateStruct, 184
 - AIRINV::FlightPeriodStruct, 193
- buildSimilarSegmentCabinSets
 - AIRINV::InventoryManager, 209
- COMMAND, 140
 - doc, 141
 - func, 140
 - name, 140
- cabin_code_p
 - AIRINV::InventoryParserHelper, 113

- AIRINV::ScheduleParserHelper, 116
- cabinCode
 - AIRINV::DCPParserHelper::DCPRuleParser, 155
- calculateAvailability
 - AIRINV::AIRINV_Master_Service, 120
 - AIRINV::AIRINV_Service, 126
 - AIRINV::InventoryHelper, 207
- cancel
 - AIRINV::AIRINV_Master_Service, 121
 - AIRINV::AIRINV_Service, 126
 - AIRINV::InventoryHelper, 207
- changeFees
 - AIRINV::DCPParserHelper::DCPRuleParser, 155
- channel
 - AIRINV::DCPParserHelper::DCPRuleParser, 155
- char_t
 - AIRINV, 106
- check
 - AIRINV::AIRINV_Master_Service, 122
 - AIRINV::AIRINV_Service, 128
- chset_t
 - AIRINV, 107
- class_code_list_p
 - AIRINV::InventoryParserHelper, 113
 - AIRINV::ScheduleParserHelper, 116
- class_code_p
 - AIRINV::InventoryParserHelper, 113
- class_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 161
- class_key
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 161
- class_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 161
- class_nego
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 161
- class_protection
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 161
- clean
 - AIRINV::FacServiceAbstract, 176
- cleanBomLayer
 - AIRINV::FacSupervisor, 179
- cleanFactory
 - AIRINV::FacSupervisor, 179
- cleanServiceLayer
 - AIRINV::FacSupervisor, 179
- ClearHistory
 - swift::SReadline, 245
- CmdAbstract, 139
- com_cd
 - readline_autocomp.hpp, 600
- com_delete
 - readline_autocomp.hpp, 600
- com_help
 - readline_autocomp.hpp, 600
- com_list
 - readline_autocomp.hpp, 599
- com_pwd
 - readline_autocomp.hpp, 599
- com_quit
 - readline_autocomp.hpp, 600
- com_rename
 - readline_autocomp.hpp, 599
- com_stat
 - readline_autocomp.hpp, 599
- com_view
 - readline_autocomp.hpp, 599
- command_generator
 - readline_autocomp.hpp, 600
- commands
 - readline_autocomp.hpp, 601
- comments
 - AIRINV::DCPParserHelper::DCPRuleParser, 153
- Connection
 - AIRINV::Connection, 141
- ConnectionShrPtr_T
 - AIRINV, 109
- content
 - AIRINV::Reply, 226
- create
 - AIRINV::FacAirinvMasterServiceContext, 171
 - AIRINV::FacAirinvServiceContext, 173
- createDirectAccesses
 - AIRINV::InventoryManager, 209
- createPickupFRAT5Curve
 - AIRINV::DefaultMap, 156
- csvDisplay
 - AIRINV::AIRINV_Master_Service, 122
 - AIRINV::AIRINV_Service, 128
- DOMESTIC
 - AIRINV::FlightTypeCode, 199
- DATADIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- DATAROOTDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- DCP
 - AIRINV::DCPParserHelper::DCPRuleParser, 155
- DCP_id
 - AIRINV::DCPParserHelper::DCPRuleParser, 153
- DCP_key
 - AIRINV::DCPParserHelper::DCPRuleParser, 153
- DCP_rule
 - AIRINV::DCPParserHelper::DCPRuleParser, 153
- DCP_rule_end
 - AIRINV::DCPParserHelper::DCPRuleParser, 153
- DCPEventStruct
 - AIRINV::DCPEventStruct, 144
- DCPFileParser
 - AIRINV::DCPEventGenerator, 142
- DCPParser

- AIRINV::DCPEventGenerator, [142](#)
- DCPParserHelper::doEndDCP
 - AIRINV::DCPEventGenerator, [142](#)
- DCPRuleFileParser
 - AIRINV::DCPRuleFileParser, [150](#)
- DCPRuleGeneration
 - AIRINV::DCPParser, [149](#)
- DCPRuleParser
 - AIRINV::DCPParserHelper::DCPRuleParser, [153](#)
- DOCDIR
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- date
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [164](#)
- date_offset
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [164](#)
- dateRangeEnd
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
- dateRangeStart
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
- day_p
 - AIRINV::DCPParserHelper, [111](#)
 - AIRINV::InventoryParserHelper, [113](#)
 - AIRINV::ScheduleParserHelper, [116](#)
- definition
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [158](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [163](#)
- DepartureDateSegmentCabinMap_T
 - AIRINV, [108](#)
- describe
 - AIRINV::BookingClassStruct, [135](#)
 - AIRINV::BucketStruct, [139](#)
 - AIRINV::DCPEventStruct, [144](#)
 - AIRINV::FareFamilyStruct, [180](#)
 - AIRINV::FlightDateStruct, [184](#)
 - AIRINV::FlightPeriodStruct, [192](#)
 - AIRINV::FlightRequestStatus, [198](#)
 - AIRINV::FlightTypeCode, [200](#)
 - AIRINV::FlightVisibilityCode, [201](#)
 - AIRINV::LegCabinStruct, [214](#)
 - AIRINV::LegStruct, [217](#)
 - AIRINV::SegmentCabinStruct, [234](#)
 - AIRINV::SegmentStruct, [237](#)
- describeKey
 - AIRINV::BomAbstract, [132](#)
- describeLabels
 - AIRINV::FlightRequestStatus, [198](#)
 - AIRINV::FlightTypeCode, [199](#)
 - AIRINV::FlightVisibilityCode, [201](#)
- describeShortKey
 - AIRINV::BomAbstract, [132](#)
- destination
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
- doEndDCP
 - AIRINV::DCPParserHelper::doEndDCP, [166](#)
- doEndFlight
 - AIRINV::ScheduleParserHelper::doEndFlight, [167](#)
- doEndFlightDate
 - AIRINV::InventoryParserHelper::doEndFlightDate,
[169](#)
- doc
 - COMMAND, [141](#)
 - doc/local/authors.doc, [612](#)
 - doc/local/codingrules.doc, [612](#)
 - doc/local/copyright.doc, [612](#)
 - doc/local/documentation.doc, [612](#)
 - doc/local/features.doc, [612](#)
 - doc/local/help_wanted.doc, [612](#)
 - doc/local/howto_release.doc, [612](#)
 - doc/local/index.doc, [612](#)
 - doc/local/installation.doc, [612](#)
 - doc/local/linking.doc, [612](#)
 - doc/local/test.doc, [612](#)
 - doc/local/users_guide.doc, [612](#)
 - doc/local/verification.doc, [612](#)
 - doc/tutorial/tutorial.doc, [612](#)
- done
 - readline_autocomp.hpp, [601](#)
- dow
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [164](#)
- dow_p
 - AIRINV::InventoryParserHelper, [113](#)
 - AIRINV::ScheduleParserHelper, [116](#)
- dupstr
 - readline_autocomp.hpp, [600](#)
- EN_FlightRequestStatus
 - AIRINV::FlightRequestStatus, [197](#)
- EN_FlightTypeCode
 - AIRINV::FlightTypeCode, [199](#)
- EN_FlightVisibilityCode
 - AIRINV::FlightVisibilityCode, [201](#)
- EXEC_PREFIX
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- enable_shared_from_this, [170](#)
- execute_line
 - readline_autocomp.hpp, [600](#)
- FFFlightPeriodFileParser
 - AIRINV::InventoryGenerator, [206](#)
- FRAT5Curve_T
 - AIRINV, [106](#)
- FacAirinvMasterServiceContext
 - AIRINV::AIRINV_Master_ServiceContext, [123](#)
 - AIRINV::FacAirinvMasterServiceContext, [171](#)
- FacAirinvServiceContext
 - AIRINV::AIRINV_ServiceContext, [129](#)
 - AIRINV::FacAirinvServiceContext, [172](#)

- FacBomAbstract
 - AIRINV::BomAbstract, [132](#)
 - AIRINV::FacBomAbstract, [174](#)
- FacServiceAbstract, [175](#)
 - AIRINV::FacServiceAbstract, [176](#)
- FacSupervisor
 - AIRINV::FacBomAbstract, [175](#)
 - AIRINV::FacSupervisor, [178](#)
- family_cabin_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [161](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [165](#)
- family_cabin_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [161](#)
- family_code_p
 - AIRINV::InventoryParserHelper, [114](#)
 - AIRINV::ScheduleParserHelper, [117](#)
- FareFamilyStruct
 - AIRINV::FareFamilyStruct, [180](#)
- FareFamilyStructList_T
 - AIRINV, [108](#)
- FileNotFoundException, [181](#)
- fileman_completion
 - readline_autocomp.hpp, [601](#)
- fill
 - AIRINV::BookingClassStruct, [135](#)
 - AIRINV::BucketStruct, [138](#)
 - AIRINV::FareFamilyStruct, [180](#)
 - AIRINV::LegCabinStruct, [214](#)
 - AIRINV::LegStruct, [217](#)
 - AIRINV::SegmentCabinStruct, [234](#)
 - AIRINV::SegmentStruct, [237](#)
- fillFromRouting
 - AIRINV::BomRootHelper, [134](#)
 - AIRINV::FlightDateHelper, [182](#)
 - AIRINV::InventoryHelper, [207](#)
 - AIRINV::SegmentDateHelper, [235](#)
- find_command
 - readline_autocomp.hpp, [600](#)
- flight_date
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- flight_date_end
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- flight_date_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- flight_key
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [163](#)
- flight_number
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [164](#)
- flight_number_p
 - AIRINV::InventoryParserHelper, [112](#)
 - AIRINV::ScheduleParserHelper, [115](#)
- flight_period
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [163](#)
- flight_period_end
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [163](#)
- flight_period_list
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [163](#)
- flight_type_code
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- flight_visibility_code
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [159](#)
- FlightDateDuplicationException
 - AIRINV::FlightDateDuplicationException, [181](#)
- FlightDateStruct
 - AIRINV::FlightDateStruct, [184](#)
- FlightPeriodFileParser
 - AIRINV::FlightPeriodFileParser, [189](#)
 - AIRINV::InventoryGenerator, [206](#)
- FlightPeriodParser
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser, [190](#)
- FlightPeriodStruct
 - AIRINV::FlightPeriodStruct, [192](#)
- FlightRequestStatus
 - AIRINV::FlightRequestStatus, [197](#)
- FlightTypeCode
 - AIRINV::FlightTypeCode, [199](#)
- FlightVisibilityCode
 - AIRINV::FlightVisibilityCode, [201](#)
- fromStream
 - AIRINV::BomAbstract, [131](#)
 - AIRINV::ServiceAbstract, [239](#)
- full_family_cabin_details
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [165](#)
- full_segment_cabin_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [160](#)
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, [164](#)
- func
 - COMMAND, [140](#)
- GROUND_HANDLING
 - AIRINV::FlightTypeCode, [199](#)
- generateDCPRules
 - AIRINV::DCPRuleFileParser, [150](#)
- generateInventories
 - AIRINV::FlightPeriodFileParser, [189](#)
 - AIRINV::ScheduleParser, [231](#)

- generic_segment
 - AIRINV::ScheduleParserHelper::FlightPeriod-Parser::definition, 165
- getAirlineListSize
 - AIRINV::DCPEventStruct, 144
- getClassCodeListSize
 - AIRINV::DCPEventStruct, 144
- getCode
 - AIRINV::FlightRequestStatus, 198
 - AIRINV::FlightTypeCode, 200
 - AIRINV::FlightVisibilityCode, 201
- getCodeLabel
 - AIRINV::FlightRequestStatus, 197
 - AIRINV::FlightTypeCode, 199
 - AIRINV::FlightVisibilityCode, 201
- getCurrentAirlineCode
 - AIRINV::DCPEventStruct, 145
- getCurrentClassCode
 - AIRINV::DCPEventStruct, 145
- getDate
 - AIRINV::DCPEventStruct, 144
 - AIRINV::FlightDateStruct, 184
 - AIRINV::FlightPeriodStruct, 192
- getFirstAirlineCode
 - AIRINV::DCPEventStruct, 144
- getFirstClassCode
 - AIRINV::DCPEventStruct, 145
- getFullSubclassCode
 - AIRINV::BookingClassStruct, 135
- GetHistory
 - swift::SReadline, 244
- getID
 - AIRINV::FacBomAbstract, 174
- getIDString
 - AIRINV::FacBomAbstract, 174, 175
- getLabel
 - AIRINV::FlightRequestStatus, 197
 - AIRINV::FlightTypeCode, 199
 - AIRINV::FlightVisibilityCode, 201
- GetLine
 - swift::SReadline, 243, 244
- getTime
 - AIRINV::DCPEventStruct, 144
 - AIRINV::FlightDateStruct, 184
 - AIRINV::FlightPeriodStruct, 192
- getYieldAndBidPrice
 - AIRINV::InventoryHelper, 207
- getwd
 - readline_autocomp.hpp, 599
- grammar, 202
- HIDDEN
 - AIRINV::FlightVisibilityCode, 201
- HTMLDIR
 - airinv-paths.hpp, 525
 - airinv-paths.hpp.in, 527
- handleRequest
 - AIRINV::RequestHandler, 228
- hasNotReachedEndAirline
 - AIRINV::DCPEventStruct, 145
- hasNotReachedEndClassCode
 - AIRINV::DCPEventStruct, 145
- hour_p
 - AIRINV::DCPParserHelper, 110
- hours_p
 - AIRINV::InventoryParserHelper, 113
 - AIRINV::ScheduleParserHelper, 116
- INTERNAL_ERROR
 - AIRINV::FlightRequestStatus, 197
- INTERNATIONAL
 - AIRINV::FlightTypeCode, 199
- INCLUDEDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- INFODIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 527
- initRMEvents
 - AIRINV::AIRINV_Service, 126
- initSnapshotAndRMEvents
 - AIRINV::AIRINV_Master_Service, 120
- initialiseAU
 - AIRINV::SegmentCabinHelper, 233
- initialize_readline
 - readline_autocomp.hpp, 601
- instance
 - AIRINV::FacAirinvMasterServiceContext, 171
 - AIRINV::FacAirinvServiceContext, 173
 - AIRINV::FacSupervisor, 178
- int1_p
 - AIRINV::DCPParserHelper, 110
 - AIRINV::InventoryParserHelper, 114
 - AIRINV::ScheduleParserHelper, 116
- int1_p_t
 - AIRINV, 106
- InventoryFileParser
 - AIRINV::InventoryFileParser, 204
- InventoryFileParsingFailedException
 - AIRINV::InventoryFileParsingFailedException, 205
- InventoryInputFileNotFoundException
 - AIRINV::InventoryInputFileNotFoundException, 208
- InventoryParser
 - AIRINV::InventoryParserHelper::InventoryParser, 211
- InventoryParserHelper::doEndFlightDate
 - AIRINV::InventoryBuilder, 204
- InventoryTestSuite, 212
 - _describeKey, 213
 - InventoryTestSuite, 212
 - InventoryTestSuite, 212
 - simpleInventory, 213
- iterateAirline
 - AIRINV::DCPEventStruct, 145
- iterateClassCode
 - AIRINV::DCPEventStruct, 146
- iterator_t

- AIRINV, 106
- jsonExport
 - AIRINV::AIRINV_Master_Service, 121
 - AIRINV::AIRINV_Service, 127
- LAST_VALUE
 - AIRINV::FlightRequestStatus, 197
 - AIRINV::FlightTypeCode, 199
 - AIRINV::FlightVisibilityCode, 201
- LIBDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- LIBEXECDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 526
- leg
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 159
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 164
- leg_cabin_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 160
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 164
- leg_cabin_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 160
- leg_details
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 160
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 164
- leg_key
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 159
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 164
- leg_list
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 159
- LegCabinStructList_T
 - AIRINV, 108
- LegStruct
 - AIRINV::LegStruct, 217
- LegStructList_T
 - AIRINV, 108
- list
 - AIRINV::AIRINV_Master_Service, 121
 - AIRINV::AIRINV_Service, 127
- list_class
 - AIRINV::DCPPParserHelper::DCPRuleParser, 156
- load
 - stdair::BomPropertyTree, 132
- LoadHistory
 - swift::SReadline, 245
- MANDIR
 - airinv-paths.hpp, 524
 - airinv-paths.hpp.in, 527
- main
 - AirInvClient.cpp, 537
 - AirInvClient_ASIO.cpp, 538
 - posix_main.cpp, 552
- minimumStay
 - AIRINV::DCPPParserHelper::DCPRuleParser, 155
- minute_p
 - AIRINV::DCPPParserHelper, 110
- minutes_p
 - AIRINV::InventoryParserHelper, 113
 - AIRINV::ScheduleParserHelper, 116
- month_p
 - AIRINV::DCPPParserHelper, 111
 - AIRINV::InventoryParserHelper, 113
 - AIRINV::ScheduleParserHelper, 116
- NORMAL
 - AIRINV::FlightVisibilityCode, 201
- NOT_FOUND
 - AIRINV::FlightRequestStatus, 197
- name
 - AIRINV::header, 203
 - COMMAND, 140
- nonRefundable
 - AIRINV::DCPPParserHelper::DCPRuleParser, 155
- noncopyable, 218
- not_to_be_parsed
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, 159
 - AIRINV::ScheduleParserHelper::FlightPeriod-
Parser::definition, 163
- OK
 - AIRINV::FlightRequestStatus, 197
- ObjectCreationgDuplicationException, 219
- operator<<
 - BomAbstract.hpp, 396
 - ServiceAbstract.hpp, 585
- operator>>
 - BomAbstract.hpp, 397
 - ServiceAbstract.hpp, 585
- operator()
 - AIRINV::DCPPParserHelper::doEndDCP, 166
 - AIRINV::DCPPParserHelper::storeAdvancePurchase,
248
 - AIRINV::DCPPParserHelper::storeAirlineCode, 253
 - AIRINV::DCPPParserHelper::storeCabinCode, 263
 - AIRINV::DCPPParserHelper::storeChangeFees, 266
 - AIRINV::DCPPParserHelper::storeChannel, 267
 - AIRINV::DCPPParserHelper::storeClass, 268
 - AIRINV::DCPPParserHelper::storeDateRangeEnd,
277
 - AIRINV::DCPPParserHelper::storeDateRangeStart,
281
 - AIRINV::DCPPParserHelper::storeDCP, 282
 - AIRINV::DCPPParserHelper::storeDCPid, 283
 - AIRINV::DCPPParserHelper::storeDestination, 285

- AIRINV::DCPParserHelper::storeEndRangeTime, [288](#)
- AIRINV::DCPParserHelper::storeMinimumStay, [315](#)
- AIRINV::DCPParserHelper::storeNonRefundable, [327](#)
- AIRINV::DCPParserHelper::storeOrigin, [335](#)
- AIRINV::DCPParserHelper::storePOS, [341](#)
- AIRINV::DCPParserHelper::storeSaturdayStay, [347](#)
- AIRINV::DCPParserHelper::storeStartRangeTime, [364](#)
- AIRINV::InventoryParserHelper::doEndFlightDate, [169](#)
- AIRINV::InventoryParserHelper::storeACP, [247](#)
- AIRINV::InventoryParserHelper::storeAirlineCode, [250](#)
- AIRINV::InventoryParserHelper::storeAU, [254](#)
- AIRINV::InventoryParserHelper::storeBoardingDate, [255](#)
- AIRINV::InventoryParserHelper::storeBoardingTime, [257](#)
- AIRINV::InventoryParserHelper::storeBookingCounter, [260](#)
- AIRINV::InventoryParserHelper::storeBucketAvailability, [262](#)
- AIRINV::InventoryParserHelper::storeClassAvailability, [269](#)
- AIRINV::InventoryParserHelper::storeClassCode, [271](#)
- AIRINV::InventoryParserHelper::storeClassETB, [274](#)
- AIRINV::InventoryParserHelper::storeCumulatedProtection, [275](#)
- AIRINV::InventoryParserHelper::storeETB, [290](#)
- AIRINV::InventoryParserHelper::storeFamilyCode, [291](#)
- AIRINV::InventoryParserHelper::storeFCClasses, [294](#)
- AIRINV::InventoryParserHelper::storeFlightDate, [297](#)
- AIRINV::InventoryParserHelper::storeFlightNumber, [299](#)
- AIRINV::InventoryParserHelper::storeFlightTypeCode, [302](#)
- AIRINV::InventoryParserHelper::storeFlightVisibilityCode, [303](#)
- AIRINV::InventoryParserHelper::storeGAV, [305](#)
- AIRINV::InventoryParserHelper::storeLegBoardingPoint, [306](#)
- AIRINV::InventoryParserHelper::storeLegCabinCode, [311](#)
- AIRINV::InventoryParserHelper::storeLegOffPoint, [313](#)
- AIRINV::InventoryParserHelper::storeNAV, [316](#)
- AIRINV::InventoryParserHelper::storeNbOfBkgs, [318](#)
- AIRINV::InventoryParserHelper::storeNbOfGroupBkgs, [320](#)
- AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs, [321](#)
- AIRINV::InventoryParserHelper::storeNbOfStaffBkgs, [323](#)
- AIRINV::InventoryParserHelper::storeNbOfWLBkgs, [324](#)
- AIRINV::InventoryParserHelper::storeNego, [326](#)
- AIRINV::InventoryParserHelper::storeNoShow, [329](#)
- AIRINV::InventoryParserHelper::storeOffDate, [330](#)
- AIRINV::InventoryParserHelper::storeOffTime, [332](#)
- AIRINV::InventoryParserHelper::storeOverbooking, [336](#)
- AIRINV::InventoryParserHelper::storeParentClassCode, [337](#)
- AIRINV::InventoryParserHelper::storeParentSubclassCode, [339](#)
- AIRINV::InventoryParserHelper::storeProtection, [342](#)
- AIRINV::InventoryParserHelper::storeRevenueAvailability, [343](#)
- AIRINV::InventoryParserHelper::storeSaleableCapacity, [345](#)
- AIRINV::InventoryParserHelper::storeSeatIndex, [348](#)
- AIRINV::InventoryParserHelper::storeSegmentAvailability, [349](#)
- AIRINV::InventoryParserHelper::storeSegmentBoardingPoint, [351](#)
- AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter, [354](#)
- AIRINV::InventoryParserHelper::storeSegmentCabinCode, [356](#)
- AIRINV::InventoryParserHelper::storeSegmentOffPoint, [359](#)
- AIRINV::InventoryParserHelper::storeSnapshotDate, [363](#)
- AIRINV::InventoryParserHelper::storeSubclassCode, [366](#)
- AIRINV::InventoryParserHelper::storeUPR, [367](#)
- AIRINV::InventoryParserHelper::storeYieldUpperRange, [369](#)
- AIRINV::ScheduleParserHelper::doEndFlight, [167](#)
- AIRINV::ScheduleParserHelper::storeAirlineCode, [251](#)
- AIRINV::ScheduleParserHelper::storeBoardingTime, [259](#)
- AIRINV::ScheduleParserHelper::storeCapacity, [264](#)
- AIRINV::ScheduleParserHelper::storeClasses, [273](#)
- AIRINV::ScheduleParserHelper::storeDateRangeEnd, [278](#)
- AIRINV::ScheduleParserHelper::storeDateRangeStart, [280](#)
- AIRINV::ScheduleParserHelper::storeDow, [286](#)
- AIRINV::ScheduleParserHelper::storeElapsedTime, [287](#)

- AIRINV::ScheduleParserHelper::storeFamilyCode, [293](#)
- AIRINV::ScheduleParserHelper::storeFCClasses, [296](#)
- AIRINV::ScheduleParserHelper::storeFlight-Number, [300](#)
- AIRINV::ScheduleParserHelper::storeLegBoarding-Point, [308](#)
- AIRINV::ScheduleParserHelper::storeLegCabin-Code, [309](#)
- AIRINV::ScheduleParserHelper::storeLegOffPoint, [312](#)
- AIRINV::ScheduleParserHelper::storeOffTime, [333](#)
- AIRINV::ScheduleParserHelper::storeSegment-BoardingPoint, [353](#)
- AIRINV::ScheduleParserHelper::storeSegment-CabinCode, [357](#)
- AIRINV::ScheduleParserHelper::storeSegmentOff-Point, [360](#)
- AIRINV::ScheduleParserHelper::storeSegment-Specificity, [361](#)
- operator=
 - swift::SKeymap, [241](#)
- optimise
 - AIRINV::AIRINV_Master_Service, [121](#)
 - AIRINV::AIRINV_Service, [127](#)
- origin
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
- PSEUDO
 - AIRINV::FlightVisibilityCode, [201](#)
- PACKAGE
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- PACKAGE_NAME
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- PACKAGE_VERSION
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- PDFDIR
 - airinv-paths.hpp, [525](#)
 - airinv-paths.hpp.in, [527](#)
- PREFIXDIR
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- parent_subclass_code
 - AIRINV::InventoryParserHelper::InventoryParser-
::definition, [161](#)
- parse
 - AIRINV::RequestParser, [229](#)
- parseAndLoad
 - AIRINV::AIRINV_Master_Service, [119](#)
 - AIRINV::AIRINV_Service, [125](#)
- parseFlightDate
 - AIRINV::Request, [227](#)
- ParserException, [219](#)
- ParserSemanticAction
 - AIRINV::DCPParserHelper::ParserSemantic-
Action, [224](#)
 - AIRINV::InventoryParserHelper::ParserSemantic-
Action, [220](#)
 - AIRINV::ScheduleParserHelper::ParserSemantic-
Action, [222](#)
- ParsingFileFailedException, [225](#)
- passenger_type_p
 - AIRINV::InventoryParserHelper, [113](#)
- position
 - AIRINV::DCPParserHelper::DCPRuleParser, [154](#)
- posix_main.cpp
 - main, [552](#)
- pt2Func
 - readline_autocomp.hpp, [599](#)
- readline_autocomp.hpp
 - com_cd, [600](#)
 - com_delete, [600](#)
 - com_help, [600](#)
 - com_list, [599](#)
 - com_pwd, [599](#)
 - com_quit, [600](#)
 - com_rename, [599](#)
 - com_stat, [599](#)
 - com_view, [599](#)
 - command_generator, [600](#)
 - commands, [601](#)
 - done, [601](#)
 - dupstr, [600](#)
 - execute_line, [600](#)
 - fileman_completion, [601](#)
 - find_command, [600](#)
 - getwd, [599](#)
 - initialize_readline, [601](#)
 - pt2Func, [599](#)
 - stripwhite, [600](#)
 - syscom, [602](#)
 - too_dangerous, [601](#)
 - valid_argument, [601](#)
 - xmalloc, [599](#)
- registerBomFactory
 - AIRINV::FacSupervisor, [178](#)
- RegisterCompletions
 - swift::SReadline, [246](#)
- registerServiceFactory
 - AIRINV::FacSupervisor, [178](#)
- repeat_p_t
 - AIRINV, [107](#)
- RequestHandler
 - AIRINV::RequestHandler, [228](#)
- RequestParser
 - AIRINV::RequestParser, [229](#)
- reset
 - AIRINV::RequestParser, [229](#)
- RootException, [229](#)
- rule_t
 - AIRINV, [106](#)
- run

- AIRINV::AirInvServer, [130](#)
- SBINDIR
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- SKeymap
 - swift::SKeymap, [240](#), [241](#)
- SReadline
 - swift::SKeymap, [241](#)
 - swift::SReadline, [243](#)
- STDAIR_SAMPLE_DIR
 - airinv-paths.hpp, [525](#)
 - airinv-paths.hpp.in, [527](#)
- SYSCONFDIR
 - airinv-paths.hpp, [524](#)
 - airinv-paths.hpp.in, [526](#)
- saturdayStay
 - AIRINV::DCPPParserHelper::DCPRuleParser, [155](#)
- save
 - stdair::BomPropertyTree, [132](#)
- SaveHistory
 - swift::SReadline, [244](#), [245](#)
- scanner_t
 - AIRINV, [106](#)
- ScheduleFileParsingFailedException
 - AIRINV::ScheduleFileParsingFailedException, [230](#)
- ScheduleInputFileNotFoundException
 - AIRINV::ScheduleInputFileNotFoundException, [231](#)
- ScheduleParser
 - AIRINV::InventoryGenerator, [206](#)
- ScheduleParserHelper::doEndFlight
 - AIRINV::InventoryGenerator, [206](#)
- second_p
 - AIRINV::DCPPParserHelper, [110](#)
- seconds_p
 - AIRINV::InventoryParserHelper, [113](#)
 - AIRINV::ScheduleParserHelper, [116](#)
- segment
 - AIRINV::DCPPParserHelper::DCPRuleParser, [155](#)
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [160](#)
- segment_cabin_details
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [161](#)
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [165](#)
- segment_cabin_key
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [160](#)
- segment_cabin_list
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [160](#)
- segment_key
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [160](#)
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [164](#)
- segment_list
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [160](#)
- segment_section
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [164](#)
- SegmentCabinStructList_T
 - AIRINV, [108](#)
- SegmentDateNotFoundException
 - AIRINV::SegmentDateNotFoundException, [236](#)
- SegmentStructList_T
 - AIRINV, [108](#)
- sell
 - AIRINV::AIRINV_Master_Service, [120](#)
 - AIRINV::AIRINV_Service, [126](#)
 - AIRINV::InventoryHelper, [207](#)
- ServiceAbstract, [238](#)
 - AIRINV::ServiceAbstract, [239](#)
- ServiceAbstract.hpp
 - operator<<, [585](#)
 - operator>>, [585](#)
- ServiceFactoryPool_T
 - AIRINV::FacSupervisor, [177](#)
- ServicePool_T
 - AIRINV::FacServiceAbstract, [176](#)
- setDefaultBidPriceVector
 - AIRINV::InventoryManager, [209](#), [210](#)
- SetKeymap
 - swift::SReadline, [246](#)
- SimilarSegmentCabinSetMap_T
 - AIRINV, [108](#)
- simpleInventory
 - InventoryTestSuite, [213](#)
- socket
 - AIRINV::Connection, [141](#)
- specific_segment_list
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [165](#)
- start
 - AIRINV::Connection, [141](#)
 - AIRINV::DCPPParserHelper::DCPRuleParser, [153](#)
 - AIRINV::InventoryParserHelper::InventoryParserHelper::definition, [158](#)
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [163](#)
- stay_duration_p
 - AIRINV::InventoryParserHelper, [113](#)
- stdair, [117](#)
 - stdair::BomPropertyTree, [132](#)
 - _airlineCode, [133](#)
 - _airportCodeList, [133](#)
 - _departureDate, [133](#)
 - _flightNumber, [133](#)
 - load, [132](#)
 - save, [132](#)
- stop
 - AIRINV::AirInvServer, [130](#)
- storeACP
 - AIRINV::InventoryParserHelper::storeACP, [247](#)

- storeAU
 - AIRINV::InventoryParserHelper::storeAU, [254](#)
- storeAdvancePurchase
 - AIRINV::DCPParserHelper::storeAdvancePurchase, [248](#)
- storeAirlineCode
 - AIRINV::DCPParserHelper::storeAirlineCode, [253](#)
 - AIRINV::InventoryParserHelper::storeAirlineCode, [250](#)
 - AIRINV::ScheduleParserHelper::storeAirlineCode, [251](#)
- storeBoardingDate
 - AIRINV::InventoryParserHelper::storeBoarding-Date, [255](#)
- storeBoardingTime
 - AIRINV::InventoryParserHelper::storeBoarding-Time, [257](#)
 - AIRINV::ScheduleParserHelper::storeBoarding-Time, [259](#)
- storeBookingCounter
 - AIRINV::InventoryParserHelper::storeBooking-Counter, [260](#)
- storeBucketAvaibility
 - AIRINV::InventoryParserHelper::storeBucket-Avaibility, [261](#)
- storeCabinCode
 - AIRINV::DCPParserHelper::storeCabinCode, [263](#)
- storeCapacity
 - AIRINV::ScheduleParserHelper::storeCapacity, [264](#)
- storeChangeFees
 - AIRINV::DCPParserHelper::storeChangeFees, [266](#)
- storeChannel
 - AIRINV::DCPParserHelper::storeChannel, [267](#)
- storeClass
 - AIRINV::DCPParserHelper::storeClass, [268](#)
- storeClassAvailability
 - AIRINV::InventoryParserHelper::storeClass-Availability, [269](#)
- storeClassCode
 - AIRINV::InventoryParserHelper::storeClassCode, [271](#)
- storeClassETB
 - AIRINV::InventoryParserHelper::storeClassETB, [274](#)
- storeClasses
 - AIRINV::ScheduleParserHelper::storeClasses, [272](#)
- storeCumulatedProtection
 - AIRINV::InventoryParserHelper::storeCumulated-Protection, [275](#)
- storeDCP
 - AIRINV::DCPParserHelper::storeDCP, [282](#)
- storeDCPId
 - AIRINV::DCPParserHelper::storeDCPId, [283](#)
- storeDateRangeEnd
 - AIRINV::DCPParserHelper::storeDateRangeEnd, [277](#)
 - AIRINV::ScheduleParserHelper::storeDateRange-End, [278](#)
- storeDateRangeStart
 - AIRINV::DCPParserHelper::storeDateRangeStart, [281](#)
 - AIRINV::ScheduleParserHelper::storeDateRange-Start, [280](#)
- storeDestination
 - AIRINV::DCPParserHelper::storeDestination, [285](#)
- storeDow
 - AIRINV::ScheduleParserHelper::storeDow, [286](#)
- storeETB
 - AIRINV::InventoryParserHelper::storeETB, [290](#)
- storeElapsedTime
 - AIRINV::ScheduleParserHelper::storeElapsed-Time, [287](#)
- storeEndRangeTime
 - AIRINV::DCPParserHelper::storeEndRangeTime, [288](#)
- storeFCClasses
 - AIRINV::InventoryParserHelper::storeFCClasses, [294](#)
 - AIRINV::ScheduleParserHelper::storeFCClasses, [296](#)
- storeFamilyCode
 - AIRINV::InventoryParserHelper::storeFamilyCode, [291](#)
 - AIRINV::ScheduleParserHelper::storeFamilyCode, [293](#)
- storeFlightDate
 - AIRINV::InventoryParserHelper::storeFlightDate, [297](#)
- storeFlightNumber
 - AIRINV::InventoryParserHelper::storeFlight-Number, [299](#)
 - AIRINV::ScheduleParserHelper::storeFlight-Number, [300](#)
- storeFlightTypeCode
 - AIRINV::InventoryParserHelper::storeFlightType-Code, [301](#)
- storeFlightVisibilityCode
 - AIRINV::InventoryParserHelper::storeFlight-VisibilityCode, [303](#)
- storeGAV
 - AIRINV::InventoryParserHelper::storeGAV, [305](#)
- storeLegBoardingPoint
 - AIRINV::InventoryParserHelper::storeLegBoarding-Point, [306](#)
 - AIRINV::ScheduleParserHelper::storeLegBoarding-Point, [308](#)
- storeLegCabinCode
 - AIRINV::InventoryParserHelper::storeLegCabin-Code, [310](#)
 - AIRINV::ScheduleParserHelper::storeLegCabin-Code, [309](#)
- storeLegOffPoint
 - AIRINV::InventoryParserHelper::storeLegOffPoint, [313](#)
 - AIRINV::ScheduleParserHelper::storeLegOffPoint,

- 312
- storeMinimumStay
 - AIRINV::DCPParserHelper::storeMinimumStay, 315
- storeNAV
 - AIRINV::InventoryParserHelper::storeNAV, 316
- storeNbOfBkgs
 - AIRINV::InventoryParserHelper::storeNbOfBkgs, 318
- storeNbOfGroupBkgs
 - AIRINV::InventoryParserHelper::storeNbOfGroupBkgs, 319
- storeNbOfPendingGroupBkgs
 - AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs, 321
- storeNbOfStaffBkgs
 - AIRINV::InventoryParserHelper::storeNbOfStaffBkgs, 323
- storeNbOfWLBkgs
 - AIRINV::InventoryParserHelper::storeNbOfWLBkgs, 324
- storeNego
 - AIRINV::InventoryParserHelper::storeNego, 326
- storeNoShow
 - AIRINV::InventoryParserHelper::storeNoShow, 328
- storeNonRefundable
 - AIRINV::DCPParserHelper::storeNonRefundable, 327
- storeOffDate
 - AIRINV::InventoryParserHelper::storeOffDate, 330
- storeOffTime
 - AIRINV::InventoryParserHelper::storeOffTime, 332
 - AIRINV::ScheduleParserHelper::storeOffTime, 333
- storeOrigin
 - AIRINV::DCPParserHelper::storeOrigin, 334
- storeOverbooking
 - AIRINV::InventoryParserHelper::storeOverbooking, 336
- storePOS
 - AIRINV::DCPParserHelper::storePOS, 340
- storeParentClassCode
 - AIRINV::InventoryParserHelper::storeParentClassCode, 337
- storeParentSubclassCode
 - AIRINV::InventoryParserHelper::storeParentSubclassCode, 339
- storeProtection
 - AIRINV::InventoryParserHelper::storeProtection, 342
- storeRevenueAvailability
 - AIRINV::InventoryParserHelper::storeRevenueAvailability, 343
- storeSaleableCapacity
 - AIRINV::InventoryParserHelper::storeSaleableCapacity, 345
- storeSaturdayStay
 - AIRINV::DCPParserHelper::storeSaturdayStay, 346
- storeSeatIndex
 - AIRINV::InventoryParserHelper::storeSeatIndex, 348
- storeSegmentAvailability
 - AIRINV::InventoryParserHelper::storeSegmentAvailability, 349
- storeSegmentBoardingPoint
 - AIRINV::InventoryParserHelper::storeSegmentBoardingPoint, 351
 - AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint, 352
- storeSegmentCabinBookingCounter
 - AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter, 354
- storeSegmentCabinCode
 - AIRINV::InventoryParserHelper::storeSegmentCabinCode, 355
 - AIRINV::ScheduleParserHelper::storeSegmentCabinCode, 357
- storeSegmentOffPoint
 - AIRINV::InventoryParserHelper::storeSegmentOffPoint, 358
 - AIRINV::ScheduleParserHelper::storeSegmentOffPoint, 360
- storeSegmentSpecificity
 - AIRINV::ScheduleParserHelper::storeSegmentSpecificity, 361
- storeSnapshotDate
 - AIRINV::InventoryParserHelper::storeSnapshotDate, 363
- storeStartRangeTime
 - AIRINV::DCPParserHelper::storeStartRangeTime, 364
- storeSubclassCode
 - AIRINV::InventoryParserHelper::storeSubclassCode, 365
- storeUPR
 - AIRINV::InventoryParserHelper::storeUPR, 367
- storeYieldUpperRange
 - AIRINV::InventoryParserHelper::storeYieldUpperRange, 369
- stripwhite
 - readline_autocomp.hpp, 600
- StructAbstract, 370
- swift, 117
- swift::SKeymap, 240
 - ~SKeymap, 240
 - Bind, 241
 - operator=, 241
 - SKeymap, 240, 241
 - SReadline, 241
 - Unbind, 241
- swift::SReadline, 242
 - ~SReadline, 243
 - ClearHistory, 245
 - GetHistory, 244
 - GetLine, 243, 244

- LoadHistory, [245](#)
- RegisterCompletions, [246](#)
- SReadline, [243](#)
- SaveHistory, [244](#), [245](#)
- SetKeymap, [246](#)
- syscom
 - readline_autocomp.hpp, [602](#)
- takeSnapshots
 - AIRINV::AIRINV_Master_Service, [121](#)
 - AIRINV::AIRINV_Service, [127](#)
 - AIRINV::GuillotineBlockHelper, [203](#)
 - AIRINV::InventoryHelper, [207](#)
- test/ Directory Reference, [102](#)
- test/airinv/ Directory Reference, [98](#)
- test/airinv/InventoryTestSuite.cpp, [612](#)
- test/airinv/InventoryTestSuite.hpp, [616](#), [617](#)
- TestFixture, [370](#)
- ThreadShrPtr_T
 - AIRINV, [108](#)
- ThreadShrPtrList_T
 - AIRINV, [108](#)
- time
 - AIRINV::DCPPParserHelper::DCPRuleParser, [154](#)
 - AIRINV::InventoryParserHelper::InventoryParser::definition, [160](#)
 - AIRINV::ScheduleParserHelper::FlightPeriodParser::definition, [164](#)
- timeRangeEnd
 - AIRINV::DCPPParserHelper::DCPRuleParser, [154](#)
- timeRangeStart
 - AIRINV::DCPPParserHelper::DCPRuleParser, [154](#)
- to_buffers
 - AIRINV::Reply, [226](#)
- toStream
 - AIRINV::BomAbstract, [131](#)
 - AIRINV::ServiceAbstract, [239](#)
- toString
 - AIRINV::BomAbstract, [132](#)
- too_dangerous
 - readline_autocomp.hpp, [601](#)
- uint1_2_p
 - AIRINV::InventoryParserHelper, [114](#)
- uint1_2_p_t
 - AIRINV, [107](#)
- uint1_3_p
 - AIRINV::InventoryParserHelper, [114](#)
- uint1_3_p_t
 - AIRINV, [107](#)
- uint1_4_p
 - AIRINV::DCPPParserHelper, [110](#)
 - AIRINV::InventoryParserHelper, [114](#)
 - AIRINV::ScheduleParserHelper, [117](#)
- uint1_4_p_t
 - AIRINV, [107](#)
- uint2_p
 - AIRINV::DCPPParserHelper, [110](#)
 - AIRINV::InventoryParserHelper, [114](#)
- AIRINV::ScheduleParserHelper, [117](#)
- uint2_p_t
 - AIRINV, [106](#)
- uint4_p
 - AIRINV::DCPPParserHelper, [110](#)
 - AIRINV::InventoryParserHelper, [114](#)
 - AIRINV::ScheduleParserHelper, [117](#)
- uint4_p_t
 - AIRINV, [107](#)
- Unbind
 - swift::SKeymap, [241](#)
- updateAUs
 - AIRINV::SegmentCabinHelper, [232](#)
- updateAvailabilities
 - AIRINV::SegmentCabinHelper, [233](#)
- updateAvailabilityPool
 - AIRINV::FlightDateHelper, [182](#)
- updateBookingControls
 - AIRINV::FlightDateHelper, [182](#)
- updateBookingControlsUsingPseudoBidPriceVector
 - AIRINV::SegmentCabinHelper, [232](#)
- updateDistanceFromElapsedTime
 - AIRINV::SegmentDateHelper, [235](#)
- updateElapsedTimeFromRouting
 - AIRINV::SegmentDateHelper, [235](#)
- updateFromReservation
 - AIRINV::SegmentCabinHelper, [232](#)
- valid_argument
 - readline_autocomp.hpp, [601](#)
- value
 - AIRINV::header, [203](#)
- xmalloc
 - readline_autocomp.hpp, [599](#)
- year_p
 - AIRINV::DCPPParserHelper, [110](#)
 - AIRINV::InventoryParserHelper, [112](#)
 - AIRINV::ScheduleParserHelper, [115](#)