

# AirTSP

## 1.01.2

Generated by Doxygen 1.6.1

Sat Jun 13 21:58:09 2015

## Contents

<b>1</b>	<b>AirTSP Documentation</b>	<b>1</b>
1.1	Getting Started . . . . .	1
1.2	AirTSP at SourceForge . . . . .	1
1.3	AirTSP Development . . . . .	1
1.4	External Libraries . . . . .	2
1.5	Support AirTSP . . . . .	2
1.6	About AirTSP . . . . .	2
<b>2</b>	<b>Configuration helper for AirTSP programs</b>	<b>2</b>
<b>3</b>	<b>People</b>	<b>3</b>
3.1	Project Admins . . . . .	3
3.2	Developers . . . . .	3
3.3	Retired Developers . . . . .	3
3.4	Contributors . . . . .	3
3.5	Distribution Maintainers . . . . .	3
<b>4</b>	<b>Coding Rules</b>	<b>3</b>
4.1	Default Naming Rules for Variables . . . . .	4
4.2	Default Naming Rules for Functions . . . . .	4
4.3	Default Naming Rules for Classes and Structures . . . . .	4
4.4	Default Naming Rules for Files . . . . .	4
4.5	Default Functionality of Classes . . . . .	4
<b>5</b>	<b>Copyright and License</b>	<b>5</b>
5.1	GNU LESSER GENERAL PUBLIC LICENSE . . . . .	5
5.1.1	Version 2.1, February 1999 . . . . .	5
5.2	Preamble . . . . .	5
5.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION . . . . .	6
5.3.1	NO WARRANTY . . . . .	10
5.3.2	END OF TERMS AND CONDITIONS . . . . .	10
5.4	How to Apply These Terms to Your New Programs . . . . .	10
<b>6</b>	<b>Documentation Rules</b>	<b>11</b>
6.1	General Rules . . . . .	11
6.2	File Header . . . . .	12
6.3	Grouping Various Parts . . . . .	12

<b>7</b>	<b>Main features</b>	<b>13</b>
7.1	Network generation . . . . .	13
7.2	Finding travel solutions . . . . .	13
7.3	Other features . . . . .	13
<b>8</b>	<b>Make a Difference</b>	<b>13</b>
<b>9</b>	<b>Make a new release</b>	<b>14</b>
9.1	Introduction . . . . .	14
9.2	Initialisation . . . . .	14
9.3	Release branch maintenance . . . . .	14
9.4	Commit and publish the release branch . . . . .	15
9.5	Create distribution packages . . . . .	15
9.6	Upload the HTML documentation to SourceForge . . . . .	15
9.7	Generate the RPM packages . . . . .	15
9.8	Update distributed change log . . . . .	16
9.9	Create the binary package, including the documentation . . . . .	16
9.10	Upload the files to SourceForge . . . . .	16
9.11	Make a new post . . . . .	16
9.12	Send an email on the announcement mailing-list . . . . .	16
<b>10</b>	<b>Installation</b>	<b>17</b>
10.1	Table of Contents . . . . .	17
10.2	Fedora/RedHat Linux distributions . . . . .	17
10.3	AirTSP Requirements . . . . .	17
10.4	Basic Installation . . . . .	18
10.5	Compilers and Options . . . . .	19
10.6	Compiling For Multiple Architectures . . . . .	19
10.7	Installation Names . . . . .	20
10.8	Optional Features . . . . .	20
10.9	Particular systems . . . . .	21
10.10	Specifying the System Type . . . . .	21
10.11	Sharing Defaults . . . . .	22
10.12	Defining Variables . . . . .	22
10.13	'cmake' Invocation . . . . .	23
<b>11</b>	<b>Linking with AirTSP</b>	<b>26</b>
11.1	Table of Contents . . . . .	26

11.2	Introduction . . . . .	27
11.3	Dependencies . . . . .	27
11.3.1	StdAir . . . . .	27
11.4	Using the pkg-config command . . . . .	27
11.5	Using the airtsp-config script . . . . .	28
11.6	M4 macro for the GNU Autotools . . . . .	28
11.7	Using AirTSP with dynamic linking . . . . .	28
<b>12</b>	<b>Test Rules</b>	<b>29</b>
12.1	The Test File . . . . .	29
12.2	The Reference File . . . . .	29
12.3	Testing IT++ Library . . . . .	29
<b>13</b>	<b>Users Guide</b>	<b>29</b>
13.1	Table of Contents . . . . .	29
13.2	Introduction . . . . .	30
13.3	Get Started . . . . .	30
13.3.1	Get the AirTSP library . . . . .	30
13.3.2	Build the AirTSP project . . . . .	30
13.3.3	Build and Run the Tests . . . . .	30
13.3.4	Install the AirTSP Project (Binaries, Documentation) . . . . .	31
13.4	Input file of AirTSP Project . . . . .	31
13.5	The schedule BOM Tree . . . . .	33
13.5.1	Build of the schedule BOM tree . . . . .	33
13.5.2	Display of the schedule BOM tree . . . . .	33
13.6	Exploring the Predefined BOM Tree . . . . .	89
13.6.1	Airline Network BOM Tree . . . . .	90
13.6.2	Airline Schedule BOM Tree . . . . .	90
13.7	Extending the BOM Tree . . . . .	90
13.8	The travel solution calculation procedure . . . . .	90
<b>14</b>	<b>Supported Systems</b>	<b>90</b>
14.1	Table of Contents . . . . .	90
14.2	Introduction . . . . .	91
14.3	AirTSP 0.2.x . . . . .	91
14.3.1	Linux Systems . . . . .	91
14.3.2	Windows Systems . . . . .	95
14.3.3	Unix Systems . . . . .	98

<b>15 AirTSP Supported Systems (Previous Releases)</b>	<b>99</b>
15.1 AirTSP 3.9.1 . . . . .	99
15.2 AirTSP 3.9.0 . . . . .	99
15.3 AirTSP 3.8.1 . . . . .	99
<b>16 Tutorials</b>	<b>99</b>
16.1 Table of Contents . . . . .	99
16.2 Preparing the AirTSP Project for Development . . . . .	99
16.3 Your first networkBuilde . . . . .	99
16.3.1 Summary of the different steps . . . . .	99
16.3.2 Result of the Batch Program . . . . .	100
16.4 Network building with an input file . . . . .	100
16.4.1 How to build a network input file? . . . . .	100
16.4.2 Building the BOM tree with an input file . . . . .	102
16.4.3 Result of the Batch Program . . . . .	102
<b>17 Command-Line Test to Demonstrate How To Test the AirTSP Project</b>	<b>103</b>
<b>18 Directory Hierarchy</b>	<b>107</b>
18.1 Directories . . . . .	107
<b>19 Namespace Index</b>	<b>108</b>
19.1 Namespace List . . . . .	108
<b>20 Class Index</b>	<b>108</b>
20.1 Class Hierarchy . . . . .	108
<b>21 Class Index</b>	<b>112</b>
21.1 Class List . . . . .	112
<b>22 File Index</b>	<b>116</b>
22.1 File List . . . . .	116
<b>23 Directory Documentation</b>	<b>119</b>
23.1 test/airtsp/ Directory Reference . . . . .	119
23.2 airtsp/ Directory Reference . . . . .	119
23.3 airtsp/basic/ Directory Reference . . . . .	119
23.4 airtsp/batches/ Directory Reference . . . . .	119
23.5 airtsp/bom/ Directory Reference . . . . .	120
23.6 airtsp/command/ Directory Reference . . . . .	120

23.7	<a href="#">airtsp/config/ Directory Reference</a>	121
23.8	<a href="#">airtsp/factory/ Directory Reference</a>	121
23.9	<a href="#">airtsp/service/ Directory Reference</a>	121
23.10	<a href="#">test/ Directory Reference</a>	121
<b>24</b>	<b>Namespace Documentation</b>	<b>122</b>
24.1	<a href="#">airtsp Namespace Reference</a>	122
24.1.1	<a href="#">Typedef Documentation</a>	122
24.1.2	<a href="#">Function Documentation</a>	123
24.1.3	<a href="#">Variable Documentation</a>	123
24.2	<a href="#">AIRTSP Namespace Reference</a>	124
24.2.1	<a href="#">Typedef Documentation</a>	127
24.2.2	<a href="#">Function Documentation</a>	131
24.2.3	<a href="#">Variable Documentation</a>	132
24.3	<a href="#">AIRTSP::OnDParserHelper Namespace Reference</a>	132
24.3.1	<a href="#">Function Documentation</a>	133
24.3.2	<a href="#">Variable Documentation</a>	134
24.4	<a href="#">AIRTSP::ScheduleParserHelper Namespace Reference</a>	135
24.4.1	<a href="#">Function Documentation</a>	136
24.4.2	<a href="#">Variable Documentation</a>	138
24.5	<a href="#">boost Namespace Reference</a>	139
24.5.1	<a href="#">Detailed Description</a>	139
24.6	<a href="#">boost::serialization Namespace Reference</a>	139
24.7	<a href="#">stdair Namespace Reference</a>	139
24.7.1	<a href="#">Detailed Description</a>	139
<b>25</b>	<b>Class Documentation</b>	<b>139</b>
25.1	<a href="#">airtsp::Airline_T Struct Reference</a>	139
25.1.1	<a href="#">Detailed Description</a>	139
25.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	140
25.1.3	<a href="#">Member Function Documentation</a>	140
25.1.4	<a href="#">Member Data Documentation</a>	140
25.2	<a href="#">AirlineScheduleTestSuite Class Reference</a>	140
25.2.1	<a href="#">Detailed Description</a>	141
25.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	141
25.2.3	<a href="#">Member Function Documentation</a>	141
25.2.4	<a href="#">Member Data Documentation</a>	141

25.3	AIRTSP::AIRTSP_Service Class Reference	142
25.3.1	Detailed Description	142
25.3.2	Constructor & Destructor Documentation	142
25.3.3	Member Function Documentation	143
25.4	AIRTSP::AIRTSP_ServiceContext Class Reference	146
25.4.1	Detailed Description	146
25.4.2	Friends And Related Function Documentation	146
25.5	BomAbstract Class Reference	147
25.6	AIRTSP::BomDisplay Class Reference	147
25.6.1	Detailed Description	147
25.6.2	Member Function Documentation	147
25.7	CmdAbstract Class Reference	148
25.8	airtsp::Date_T Struct Reference	149
25.8.1	Detailed Description	150
25.8.2	Constructor & Destructor Documentation	150
25.8.3	Member Function Documentation	150
25.8.4	Member Data Documentation	150
25.9	AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference	151
25.9.1	Detailed Description	152
25.9.2	Constructor & Destructor Documentation	152
25.9.3	Member Function Documentation	153
25.9.4	Member Data Documentation	153
25.10	airtsp::SearchStringParser::definition< ScannerT > Struct Template Reference	158
25.10.1	Detailed Description	159
25.10.2	Constructor & Destructor Documentation	159
25.10.3	Member Function Documentation	159
25.10.4	Member Data Documentation	159
25.11	AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT > Struct Template Reference	162
25.11.1	Detailed Description	163
25.11.2	Constructor & Destructor Documentation	163
25.11.3	Member Function Documentation	163
25.11.4	Member Data Documentation	164
25.12	AIRTSP::ScheduleParserHelper::doEndFlight Struct Reference	165
25.12.1	Detailed Description	165
25.12.2	Constructor & Destructor Documentation	165
25.12.3	Member Function Documentation	166

25.12.4 Member Data Documentation . . . . .	166
25.13AIRTSP::OnDParserHelper::doEndOnD Struct Reference . . . . .	166
25.13.1 Detailed Description . . . . .	167
25.13.2 Constructor & Destructor Documentation . . . . .	167
25.13.3 Member Function Documentation . . . . .	167
25.13.4 Member Data Documentation . . . . .	167
25.14AIRTSP::FacAIRTSPServiceContext Class Reference . . . . .	168
25.14.1 Detailed Description . . . . .	168
25.14.2 Constructor & Destructor Documentation . . . . .	169
25.14.3 Member Function Documentation . . . . .	169
25.15AIRTSP::FacServiceAbstract Class Reference . . . . .	170
25.15.1 Detailed Description . . . . .	170
25.15.2 Member Typedef Documentation . . . . .	170
25.15.3 Constructor & Destructor Documentation . . . . .	170
25.15.4 Member Function Documentation . . . . .	171
25.15.5 Member Data Documentation . . . . .	171
25.16FacServiceAbstract Class Reference . . . . .	171
25.17AIRTSP::FareFamilyStruct Struct Reference . . . . .	171
25.17.1 Detailed Description . . . . .	172
25.17.2 Constructor & Destructor Documentation . . . . .	172
25.17.3 Member Function Documentation . . . . .	172
25.17.4 Member Data Documentation . . . . .	172
25.18FileNotFoundException Class Reference . . . . .	173
25.19AIRTSP::FlagSaver Struct Reference . . . . .	173
25.19.1 Detailed Description . . . . .	173
25.19.2 Constructor & Destructor Documentation . . . . .	174
25.20AIRTSP::FlightPeriodFileParser Class Reference . . . . .	174
25.20.1 Detailed Description . . . . .	174
25.20.2 Constructor & Destructor Documentation . . . . .	174
25.20.3 Member Function Documentation . . . . .	175
25.21AIRTSP::ScheduleParserHelper::FlightPeriodParser Struct Reference . . . . .	175
25.21.1 Detailed Description . . . . .	175
25.21.2 Constructor & Destructor Documentation . . . . .	176
25.21.3 Member Data Documentation . . . . .	176
25.22AIRTSP::FlightPeriodStruct Struct Reference . . . . .	176
25.22.1 Detailed Description . . . . .	177



25.22.2 Constructor & Destructor Documentation . . . . .	178
25.22.3 Member Function Documentation . . . . .	178
25.22.4 Member Data Documentation . . . . .	180
25.23grammar Class Reference . . . . .	184
25.24AIRTSP::InventoryGenerator Class Reference . . . . .	184
25.24.1 Detailed Description . . . . .	184
25.24.2 Friends And Related Function Documentation . . . . .	184
25.25KeyAbstract Class Reference . . . . .	185
25.26AIRTSP::LegCabinStruct Struct Reference . . . . .	185
25.26.1 Detailed Description . . . . .	186
25.26.2 Member Function Documentation . . . . .	186
25.26.3 Member Data Documentation . . . . .	186
25.27AIRTSP::LegStruct Struct Reference . . . . .	187
25.27.1 Detailed Description . . . . .	187
25.27.2 Constructor & Destructor Documentation . . . . .	187
25.27.3 Member Function Documentation . . . . .	188
25.27.4 Member Data Documentation . . . . .	188
25.28AIRTSP::OnInputFileNotFoundException Class Reference . . . . .	190
25.28.1 Detailed Description . . . . .	190
25.28.2 Constructor & Destructor Documentation . . . . .	190
25.29AIRTSP::OnDParser Class Reference . . . . .	190
25.29.1 Detailed Description . . . . .	191
25.29.2 Member Function Documentation . . . . .	191
25.30AIRTSP::OnDParserHelper::OnDParser Struct Reference . . . . .	191
25.30.1 Detailed Description . . . . .	192
25.30.2 Constructor & Destructor Documentation . . . . .	192
25.30.3 Member Data Documentation . . . . .	192
25.31AIRTSP::OnDPeriodFileParser Class Reference . . . . .	192
25.31.1 Detailed Description . . . . .	193
25.31.2 Constructor & Destructor Documentation . . . . .	193
25.31.3 Member Function Documentation . . . . .	193
25.32AIRTSP::OnDPeriodGenerator Class Reference . . . . .	193
25.32.1 Detailed Description . . . . .	194
25.32.2 Friends And Related Function Documentation . . . . .	194
25.33AIRTSP::OnDPeriodStruct Struct Reference . . . . .	194
25.33.1 Detailed Description . . . . .	195

25.33.2 Constructor & Destructor Documentation . . . . .	195
25.33.3 Member Function Documentation . . . . .	196
25.33.4 Member Data Documentation . . . . .	196
25.34AIRTSP::OriginDestinationSet Class Reference . . . . .	199
25.34.1 Detailed Description . . . . .	200
25.34.2 Member Typedef Documentation . . . . .	201
25.34.3 Constructor & Destructor Documentation . . . . .	201
25.34.4 Member Function Documentation . . . . .	201
25.34.5 Friends And Related Function Documentation . . . . .	203
25.34.6 Member Data Documentation . . . . .	203
25.35AIRTSP::OriginDestinationSetKey Struct Reference . . . . .	204
25.35.1 Detailed Description . . . . .	204
25.35.2 Constructor & Destructor Documentation . . . . .	204
25.35.3 Member Function Documentation . . . . .	205
25.35.4 Friends And Related Function Documentation . . . . .	206
25.36ParserException Class Reference . . . . .	206
25.37AIRTSP::OnDParserHelper::ParserSemanticAction Struct Reference . . . . .	206
25.37.1 Detailed Description . . . . .	207
25.37.2 Constructor & Destructor Documentation . . . . .	207
25.37.3 Member Data Documentation . . . . .	207
25.38AIRTSP::ScheduleParserHelper::ParserSemanticAction Struct Reference . . . . .	208
25.38.1 Detailed Description . . . . .	209
25.38.2 Constructor & Destructor Documentation . . . . .	209
25.38.3 Member Data Documentation . . . . .	209
25.39airtsp::Passenger_T Struct Reference . . . . .	209
25.39.1 Detailed Description . . . . .	210
25.39.2 Member Enumeration Documentation . . . . .	210
25.39.3 Constructor & Destructor Documentation . . . . .	210
25.39.4 Member Function Documentation . . . . .	211
25.39.5 Member Data Documentation . . . . .	211
25.40airtsp::Place_T Struct Reference . . . . .	211
25.40.1 Detailed Description . . . . .	212
25.40.2 Constructor & Destructor Documentation . . . . .	212
25.40.3 Member Function Documentation . . . . .	212
25.40.4 Member Data Documentation . . . . .	212
25.41AIRTSP::ReachableUniverse Class Reference . . . . .	213

25.41.1 Detailed Description . . . . .	214
25.41.2 Member Typedef Documentation . . . . .	214
25.41.3 Constructor & Destructor Documentation . . . . .	214
25.41.4 Member Function Documentation . . . . .	214
25.41.5 Friends And Related Function Documentation . . . . .	216
25.41.6 Member Data Documentation . . . . .	216
25.42AIRTSP::ReachableUniverseKey Struct Reference . . . . .	217
25.42.1 Detailed Description . . . . .	218
25.42.2 Constructor & Destructor Documentation . . . . .	218
25.42.3 Member Function Documentation . . . . .	218
25.42.4 Friends And Related Function Documentation . . . . .	219
25.43AIRTSP::ScheduleInputFileNotFoundException Class Reference . . . . .	220
25.43.1 Detailed Description . . . . .	220
25.43.2 Constructor & Destructor Documentation . . . . .	220
25.44AIRTSP::ScheduleParser Class Reference . . . . .	220
25.44.1 Detailed Description . . . . .	221
25.44.2 Member Function Documentation . . . . .	221
25.45airtsp::SearchString_T Struct Reference . . . . .	221
25.45.1 Detailed Description . . . . .	222
25.45.2 Constructor & Destructor Documentation . . . . .	222
25.45.3 Member Function Documentation . . . . .	222
25.45.4 Member Data Documentation . . . . .	222
25.46airtsp::SearchStringParser Struct Reference . . . . .	223
25.46.1 Detailed Description . . . . .	224
25.46.2 Constructor & Destructor Documentation . . . . .	224
25.46.3 Member Data Documentation . . . . .	224
25.47AIRTSP::SegmentCabinStruct Struct Reference . . . . .	224
25.47.1 Detailed Description . . . . .	225
25.47.2 Member Function Documentation . . . . .	225
25.47.3 Member Data Documentation . . . . .	225
25.48AIRTSP::SegmentDateNotFoundException Class Reference . . . . .	226
25.48.1 Detailed Description . . . . .	227
25.48.2 Constructor & Destructor Documentation . . . . .	227
25.49AIRTSP::SegmentPathGenerator Class Reference . . . . .	227
25.49.1 Detailed Description . . . . .	227
25.49.2 Member Function Documentation . . . . .	228

25.50	AIRTSP::SegmentPathPeriod Class Reference	228
25.50.1	Detailed Description	229
25.50.2	Member Typedef Documentation	229
25.50.3	Constructor & Destructor Documentation	229
25.50.4	Member Function Documentation	230
25.50.5	Friends And Related Function Documentation	233
25.50.6	Member Data Documentation	234
25.51	AIRTSP::SegmentPathPeriodKey Struct Reference	234
25.51.1	Detailed Description	235
25.51.2	Constructor & Destructor Documentation	236
25.51.3	Member Function Documentation	236
25.51.4	Friends And Related Function Documentation	239
25.52	AIRTSP::SegmentPathProvider Class Reference	239
25.52.1	Detailed Description	239
25.52.2	Friends And Related Function Documentation	240
25.53	AIRTSP::SegmentPeriodHelper Class Reference	240
25.53.1	Detailed Description	240
25.53.2	Member Function Documentation	240
25.54	AIRTSP::SegmentStruct Struct Reference	241
25.54.1	Detailed Description	241
25.54.2	Member Function Documentation	241
25.54.3	Member Data Documentation	242
25.55	ServiceAbstract Class Reference	243
25.56	AIRTSP::ServiceAbstract Class Reference	243
25.56.1	Detailed Description	244
25.56.2	Constructor & Destructor Documentation	244
25.56.3	Member Function Documentation	244
25.57	AIRTSP::Simulator Class Reference	244
25.57.1	Detailed Description	245
25.57.2	Member Function Documentation	245
25.58	airtsp::store_adult_passenger_type Struct Reference	245
25.58.1	Detailed Description	245
25.58.2	Constructor & Destructor Documentation	246
25.58.3	Member Function Documentation	246
25.58.4	Member Data Documentation	246
25.59	airtsp::store_airline_code Struct Reference	246

25.59.1 Detailed Description . . . . .	246
25.59.2 Constructor & Destructor Documentation . . . . .	247
25.59.3 Member Function Documentation . . . . .	247
25.59.4 Member Data Documentation . . . . .	247
25.60airtsp::store_airline_name Struct Reference . . . . .	247
25.60.1 Detailed Description . . . . .	247
25.60.2 Constructor & Destructor Documentation . . . . .	248
25.60.3 Member Function Documentation . . . . .	248
25.60.4 Member Data Documentation . . . . .	248
25.61airtsp::store_airline_sign Struct Reference . . . . .	248
25.61.1 Detailed Description . . . . .	248
25.61.2 Constructor & Destructor Documentation . . . . .	249
25.61.3 Member Function Documentation . . . . .	249
25.61.4 Member Data Documentation . . . . .	249
25.62airtsp::store_child_passenger_type Struct Reference . . . . .	249
25.62.1 Detailed Description . . . . .	249
25.62.2 Constructor & Destructor Documentation . . . . .	250
25.62.3 Member Function Documentation . . . . .	250
25.62.4 Member Data Documentation . . . . .	250
25.63airtsp::store_date Struct Reference . . . . .	250
25.63.1 Detailed Description . . . . .	250
25.63.2 Constructor & Destructor Documentation . . . . .	251
25.63.3 Member Function Documentation . . . . .	251
25.63.4 Member Data Documentation . . . . .	251
25.64airtsp::store_passenger_number Struct Reference . . . . .	251
25.64.1 Detailed Description . . . . .	251
25.64.2 Constructor & Destructor Documentation . . . . .	252
25.64.3 Member Function Documentation . . . . .	252
25.64.4 Member Data Documentation . . . . .	252
25.65airtsp::store_pet_passenger_type Struct Reference . . . . .	252
25.65.1 Detailed Description . . . . .	252
25.65.2 Constructor & Destructor Documentation . . . . .	253
25.65.3 Member Function Documentation . . . . .	253
25.65.4 Member Data Documentation . . . . .	253
25.66airtsp::store_place_element Struct Reference . . . . .	253
25.66.1 Detailed Description . . . . .	253

25.66.2 Constructor & Destructor Documentation . . . . .	254
25.66.3 Member Function Documentation . . . . .	254
25.66.4 Member Data Documentation . . . . .	254
25.67 AIRTSP::OnDParserHelper::storeAirlineCode Struct Reference . . . . .	254
25.67.1 Detailed Description . . . . .	255
25.67.2 Constructor & Destructor Documentation . . . . .	255
25.67.3 Member Function Documentation . . . . .	255
25.67.4 Member Data Documentation . . . . .	255
25.68 AIRTSP::ScheduleParserHelper::storeAirlineCode Struct Reference . . . . .	255
25.68.1 Detailed Description . . . . .	256
25.68.2 Constructor & Destructor Documentation . . . . .	256
25.68.3 Member Function Documentation . . . . .	256
25.68.4 Member Data Documentation . . . . .	256
25.69 AIRTSP::ScheduleParserHelper::storeBoardingTime Struct Reference . . . . .	257
25.69.1 Detailed Description . . . . .	257
25.69.2 Constructor & Destructor Documentation . . . . .	258
25.69.3 Member Function Documentation . . . . .	258
25.69.4 Member Data Documentation . . . . .	258
25.70 AIRTSP::ScheduleParserHelper::storeCapacity Struct Reference . . . . .	258
25.70.1 Detailed Description . . . . .	259
25.70.2 Constructor & Destructor Documentation . . . . .	259
25.70.3 Member Function Documentation . . . . .	259
25.70.4 Member Data Documentation . . . . .	259
25.71 AIRTSP::OnDParserHelper::storeClassCode Struct Reference . . . . .	260
25.71.1 Detailed Description . . . . .	260
25.71.2 Constructor & Destructor Documentation . . . . .	261
25.71.3 Member Function Documentation . . . . .	261
25.71.4 Member Data Documentation . . . . .	261
25.72 AIRTSP::ScheduleParserHelper::storeClasses Struct Reference . . . . .	261
25.72.1 Detailed Description . . . . .	262
25.72.2 Constructor & Destructor Documentation . . . . .	262
25.72.3 Member Function Documentation . . . . .	262
25.72.4 Member Data Documentation . . . . .	262
25.73 AIRTSP::ScheduleParserHelper::storeDateRangeEnd Struct Reference . . . . .	263
25.73.1 Detailed Description . . . . .	263
25.73.2 Constructor & Destructor Documentation . . . . .	263

25.73.3 Member Function Documentation . . . . .	264
25.73.4 Member Data Documentation . . . . .	264
25.74 AIRTSP::OnDParserHelper::storeDateRangeEnd Struct Reference . . . . .	264
25.74.1 Detailed Description . . . . .	265
25.74.2 Constructor & Destructor Documentation . . . . .	265
25.74.3 Member Function Documentation . . . . .	265
25.74.4 Member Data Documentation . . . . .	265
25.75 AIRTSP::OnDParserHelper::storeDateRangeStart Struct Reference . . . . .	266
25.75.1 Detailed Description . . . . .	266
25.75.2 Constructor & Destructor Documentation . . . . .	266
25.75.3 Member Function Documentation . . . . .	266
25.75.4 Member Data Documentation . . . . .	267
25.76 AIRTSP::ScheduleParserHelper::storeDateRangeStart Struct Reference . . . . .	267
25.76.1 Detailed Description . . . . .	267
25.76.2 Constructor & Destructor Documentation . . . . .	267
25.76.3 Member Function Documentation . . . . .	268
25.76.4 Member Data Documentation . . . . .	268
25.77 AIRTSP::OnDParserHelper::storeDestination Struct Reference . . . . .	268
25.77.1 Detailed Description . . . . .	269
25.77.2 Constructor & Destructor Documentation . . . . .	269
25.77.3 Member Function Documentation . . . . .	269
25.77.4 Member Data Documentation . . . . .	269
25.78 AIRTSP::ScheduleParserHelper::storeDow Struct Reference . . . . .	270
25.78.1 Detailed Description . . . . .	270
25.78.2 Constructor & Destructor Documentation . . . . .	270
25.78.3 Member Function Documentation . . . . .	270
25.78.4 Member Data Documentation . . . . .	271
25.79 AIRTSP::ScheduleParserHelper::storeElapsedTime Struct Reference . . . . .	271
25.79.1 Detailed Description . . . . .	271
25.79.2 Constructor & Destructor Documentation . . . . .	272
25.79.3 Member Function Documentation . . . . .	272
25.79.4 Member Data Documentation . . . . .	272
25.80 AIRTSP::OnDParserHelper::storeEndRangeTime Struct Reference . . . . .	273
25.80.1 Detailed Description . . . . .	273
25.80.2 Constructor & Destructor Documentation . . . . .	273
25.80.3 Member Function Documentation . . . . .	273

25.80.4 Member Data Documentation . . . . .	274
25.81 AIRTSP::ScheduleParserHelper::storeFamilyCode Struct Reference . . . . .	274
25.81.1 Detailed Description . . . . .	274
25.81.2 Constructor & Destructor Documentation . . . . .	274
25.81.3 Member Function Documentation . . . . .	275
25.81.4 Member Data Documentation . . . . .	275
25.82 AIRTSP::ScheduleParserHelper::storeFCClasses Struct Reference . . . . .	275
25.82.1 Detailed Description . . . . .	276
25.82.2 Constructor & Destructor Documentation . . . . .	276
25.82.3 Member Function Documentation . . . . .	276
25.82.4 Member Data Documentation . . . . .	276
25.83 AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey Struct Reference . . . . .	277
25.83.1 Detailed Description . . . . .	277
25.83.2 Constructor & Destructor Documentation . . . . .	277
25.83.3 Member Function Documentation . . . . .	278
25.83.4 Member Data Documentation . . . . .	278
25.84 AIRTSP::ScheduleParserHelper::storeFlightNumber Struct Reference . . . . .	278
25.84.1 Detailed Description . . . . .	279
25.84.2 Constructor & Destructor Documentation . . . . .	279
25.84.3 Member Function Documentation . . . . .	279
25.84.4 Member Data Documentation . . . . .	279
25.85 AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey Struct Reference . . . . .	280
25.85.1 Detailed Description . . . . .	280
25.85.2 Constructor & Destructor Documentation . . . . .	280
25.85.3 Member Function Documentation . . . . .	281
25.85.4 Member Data Documentation . . . . .	281
25.86 AIRTSP::ScheduleParserHelper::storeLegBoardingPoint Struct Reference . . . . .	281
25.86.1 Detailed Description . . . . .	282
25.86.2 Constructor & Destructor Documentation . . . . .	282
25.86.3 Member Function Documentation . . . . .	282
25.86.4 Member Data Documentation . . . . .	282
25.87 AIRTSP::ScheduleParserHelper::storeLegCabinCode Struct Reference . . . . .	283
25.87.1 Detailed Description . . . . .	283
25.87.2 Constructor & Destructor Documentation . . . . .	283
25.87.3 Member Function Documentation . . . . .	284
25.87.4 Member Data Documentation . . . . .	284



25.88	AIRTSP::ScheduleParserHelper::storeLegOffPoint Struct Reference	284
25.88.1	Detailed Description	285
25.88.2	Constructor & Destructor Documentation	285
25.88.3	Member Function Documentation	285
25.88.4	Member Data Documentation	285
25.89	AIRTSP::ScheduleParserHelper::storeOffTime Struct Reference	286
25.89.1	Detailed Description	286
25.89.2	Constructor & Destructor Documentation	286
25.89.3	Member Function Documentation	287
25.89.4	Member Data Documentation	287
25.90	AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode Struct Reference	287
25.90.1	Detailed Description	288
25.90.2	Constructor & Destructor Documentation	288
25.90.3	Member Function Documentation	288
25.90.4	Member Data Documentation	288
25.91	AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber Struct Reference	289
25.91.1	Detailed Description	289
25.91.2	Constructor & Destructor Documentation	289
25.91.3	Member Function Documentation	290
25.91.4	Member Data Documentation	290
25.92	AIRTSP::OnDParserHelper::storeOrigin Struct Reference	290
25.92.1	Detailed Description	291
25.92.2	Constructor & Destructor Documentation	291
25.92.3	Member Function Documentation	291
25.92.4	Member Data Documentation	291
25.93	AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference	292
25.93.1	Detailed Description	292
25.93.2	Constructor & Destructor Documentation	292
25.93.3	Member Function Documentation	292
25.93.4	Member Data Documentation	293
25.94	AIRTSP::ScheduleParserHelper::storeSegmentCabinCode Struct Reference	293
25.94.1	Detailed Description	294
25.94.2	Constructor & Destructor Documentation	294
25.94.3	Member Function Documentation	294
25.94.4	Member Data Documentation	294
25.95	AIRTSP::ScheduleParserHelper::storeSegmentOffPoint Struct Reference	295

25.95.1 Detailed Description . . . . .	295
25.95.2 Constructor & Destructor Documentation . . . . .	295
25.95.3 Member Function Documentation . . . . .	295
25.95.4 Member Data Documentation . . . . .	296
25.96 AIRTSP::ScheduleParserHelper::storeSegmentSpecificity Struct Reference . . . . .	296
25.96.1 Detailed Description . . . . .	297
25.96.2 Constructor & Destructor Documentation . . . . .	297
25.96.3 Member Function Documentation . . . . .	297
25.96.4 Member Data Documentation . . . . .	297
25.97 AIRTSP::OnDParserHelper::storeStartRangeTime Struct Reference . . . . .	298
25.97.1 Detailed Description . . . . .	298
25.97.2 Constructor & Destructor Documentation . . . . .	298
25.97.3 Member Function Documentation . . . . .	298
25.97.4 Member Data Documentation . . . . .	299
25.98 StructAbstract Class Reference . . . . .	299
25.99 TestFixture Class Reference . . . . .	299
25.100 AIRTSP::TravelSolutionParser Class Reference . . . . .	300
25.100.1 Detailed Description . . . . .	300
25.100.2 Member Function Documentation . . . . .	300
<b>26 File Documentation . . . . .</b>	<b>300</b>
26.1 airtsp/AIRTSP_Service.hpp File Reference . . . . .	300
26.2 AIRTSP_Service.hpp . . . . .	302
26.3 airtsp/AIRTSP_Types.hpp File Reference . . . . .	304
26.4 AIRTSP_Types.hpp . . . . .	305
26.5 airtsp/basic/BasConst.cpp File Reference . . . . .	306
26.6 BasConst.cpp . . . . .	307
26.7 airtsp/basic/BasConst_AIRTSP_Service.hpp File Reference . . . . .	308
26.8 BasConst_AIRTSP_Service.hpp . . . . .	309
26.9 airtsp/basic/BasConst_General.hpp File Reference . . . . .	310
26.10 BasConst_General.hpp . . . . .	311
26.11 airtsp/basic/BasParserTypes.hpp File Reference . . . . .	312
26.12 BasParserTypes.hpp . . . . .	313
26.13 airtsp/batches/airtsp.cpp File Reference . . . . .	314
26.13.1 Typedef Documentation . . . . .	315
26.13.2 Function Documentation . . . . .	315
26.13.3 Variable Documentation . . . . .	316

26.14	<a href="#">airtsp.cpp</a>	317
26.15	<a href="#">airtsp/batches/BookingRequestParser.cpp File Reference</a>	324
26.15.1	<a href="#">Define Documentation</a>	325
26.15.2	<a href="#">Typedef Documentation</a>	325
26.16	<a href="#">BookingRequestParser.cpp</a>	327
26.17	<a href="#">airtsp/batches/BookingRequestParser.hpp File Reference</a>	333
26.18	<a href="#">BookingRequestParser.hpp</a>	334
26.19	<a href="#">airtsp/bom/AirportList.hpp File Reference</a>	337
26.20	<a href="#">AirportList.hpp</a>	338
26.21	<a href="#">airtsp/bom/BomDisplay.cpp File Reference</a>	339
26.22	<a href="#">BomDisplay.cpp</a>	340
26.23	<a href="#">airtsp/bom/BomDisplay.hpp File Reference</a>	342
26.24	<a href="#">BomDisplay.hpp</a>	343
26.25	<a href="#">airtsp/bom/FareFamilyStruct.cpp File Reference</a>	344
26.26	<a href="#">FareFamilyStruct.cpp</a>	345
26.27	<a href="#">airtsp/bom/FareFamilyStruct.hpp File Reference</a>	346
26.28	<a href="#">FareFamilyStruct.hpp</a>	347
26.29	<a href="#">airtsp/bom/FlightPeriodStruct.cpp File Reference</a>	348
26.30	<a href="#">FlightPeriodStruct.cpp</a>	349
26.31	<a href="#">airtsp/bom/FlightPeriodStruct.hpp File Reference</a>	353
26.32	<a href="#">FlightPeriodStruct.hpp</a>	354
26.33	<a href="#">airtsp/bom/LegCabinStruct.cpp File Reference</a>	356
26.34	<a href="#">LegCabinStruct.cpp</a>	357
26.35	<a href="#">airtsp/bom/LegCabinStruct.hpp File Reference</a>	358
26.36	<a href="#">LegCabinStruct.hpp</a>	359
26.37	<a href="#">airtsp/bom/LegStruct.cpp File Reference</a>	360
26.38	<a href="#">LegStruct.cpp</a>	361
26.39	<a href="#">airtsp/bom/LegStruct.hpp File Reference</a>	363
26.40	<a href="#">LegStruct.hpp</a>	364
26.41	<a href="#">airtsp/bom/OnDPeriodStruct.cpp File Reference</a>	365
26.42	<a href="#">OnDPeriodStruct.cpp</a>	366
26.43	<a href="#">airtsp/bom/OnDPeriodStruct.hpp File Reference</a>	368
26.44	<a href="#">OnDPeriodStruct.hpp</a>	369
26.45	<a href="#">airtsp/bom/OriginDestinationSet.cpp File Reference</a>	370
26.46	<a href="#">OriginDestinationSet.cpp</a>	371
26.47	<a href="#">airtsp/bom/OriginDestinationSet.hpp File Reference</a>	373

26.48OriginDestinationSet.hpp . . . . .	374
26.49airtsp/bom/OriginDestinationSetKey.cpp File Reference . . . . .	376
26.50OriginDestinationSetKey.cpp . . . . .	377
26.51airtsp/bom/OriginDestinationSetKey.hpp File Reference . . . . .	379
26.52OriginDestinationSetKey.hpp . . . . .	380
26.53airtsp/bom/OriginDestinationSetTypes.hpp File Reference . . . . .	382
26.54OriginDestinationSetTypes.hpp . . . . .	383
26.55airtsp/bom/ReachableUniverse.cpp File Reference . . . . .	384
26.56ReachableUniverse.cpp . . . . .	385
26.57airtsp/bom/ReachableUniverse.hpp File Reference . . . . .	387
26.58ReachableUniverse.hpp . . . . .	388
26.59airtsp/bom/ReachableUniverseKey.cpp File Reference . . . . .	390
26.60ReachableUniverseKey.cpp . . . . .	391
26.61airtsp/bom/ReachableUniverseKey.hpp File Reference . . . . .	393
26.62ReachableUniverseKey.hpp . . . . .	394
26.63airtsp/bom/ReachableUniverseTypes.hpp File Reference . . . . .	396
26.64ReachableUniverseTypes.hpp . . . . .	397
26.65airtsp/bom/SegmentCabinStruct.cpp File Reference . . . . .	398
26.66SegmentCabinStruct.cpp . . . . .	399
26.67airtsp/bom/SegmentCabinStruct.hpp File Reference . . . . .	400
26.68SegmentCabinStruct.hpp . . . . .	401
26.69airtsp/bom/SegmentPathPeriod.cpp File Reference . . . . .	402
26.70SegmentPathPeriod.cpp . . . . .	403
26.71airtsp/bom/SegmentPathPeriod.hpp File Reference . . . . .	408
26.72SegmentPathPeriod.hpp . . . . .	409
26.73airtsp/bom/SegmentPathPeriodKey.cpp File Reference . . . . .	412
26.74SegmentPathPeriodKey.cpp . . . . .	413
26.75airtsp/bom/SegmentPathPeriodKey.hpp File Reference . . . . .	415
26.76SegmentPathPeriodKey.hpp . . . . .	416
26.77airtsp/bom/SegmentPathPeriodTypes.hpp File Reference . . . . .	419
26.78SegmentPathPeriodTypes.hpp . . . . .	420
26.79airtsp/bom/SegmentPeriodHelper.cpp File Reference . . . . .	421
26.80SegmentPeriodHelper.cpp . . . . .	422
26.81airtsp/bom/SegmentPeriodHelper.hpp File Reference . . . . .	424
26.82SegmentPeriodHelper.hpp . . . . .	425
26.83airtsp/bom/SegmentStruct.cpp File Reference . . . . .	426

26.84SegmentStruct.cpp . . . . .	427
26.85airtsp/bom/SegmentStruct.hpp File Reference . . . . .	428
26.86SegmentStruct.hpp . . . . .	429
26.87airtsp/command/InventoryGenerator.cpp File Reference . . . . .	430
26.88InventoryGenerator.cpp . . . . .	431
26.89airtsp/command/InventoryGenerator.hpp File Reference . . . . .	433
26.90InventoryGenerator.hpp . . . . .	434
26.91airtsp/command/OnDParser.cpp File Reference . . . . .	435
26.92OnDParser.cpp . . . . .	436
26.93airtsp/command/OnDParser.hpp File Reference . . . . .	437
26.94OnDParser.hpp . . . . .	438
26.95airtsp/command/OnDParserHelper.cpp File Reference . . . . .	439
26.96OnDParserHelper.cpp . . . . .	440
26.97airtsp/command/OnDParserHelper.hpp File Reference . . . . .	446
26.98OnDParserHelper.hpp . . . . .	447
26.99airtsp/command/OnDPeriodGenerator.cpp File Reference . . . . .	450
26.100OnDPeriodGenerator.cpp . . . . .	451
26.101airtsp/command/OnDPeriodGenerator.hpp File Reference . . . . .	452
26.102OnDPeriodGenerator.hpp . . . . .	453
26.103airtsp/command/ScheduleParser.cpp File Reference . . . . .	454
26.104ScheduleParser.cpp . . . . .	455
26.105airtsp/command/ScheduleParser.hpp File Reference . . . . .	456
26.106ScheduleParser.hpp . . . . .	457
26.107airtsp/command/ScheduleParserHelper.cpp File Reference . . . . .	458
26.108ScheduleParserHelper.cpp . . . . .	459
26.109airtsp/command/ScheduleParserHelper.hpp File Reference . . . . .	471
26.110ScheduleParserHelper.hpp . . . . .	472
26.111airtsp/command/SegmentPathGenerator.cpp File Reference . . . . .	476
26.112SegmentPathGenerator.cpp . . . . .	477
26.113airtsp/command/SegmentPathGenerator.hpp File Reference . . . . .	484
26.114SegmentPathGenerator.hpp . . . . .	485
26.115airtsp/command/SegmentPathProvider.cpp File Reference . . . . .	486
26.116SegmentPathProvider.cpp . . . . .	487
26.117airtsp/command/SegmentPathProvider.hpp File Reference . . . . .	490
26.118SegmentPathProvider.hpp . . . . .	491
26.119airtsp/command/Simulator.cpp File Reference . . . . .	492

26.128imulator.cpp . . . . .	493
26.12airtsp/command/Simulator.hpp File Reference . . . . .	494
26.128imulator.hpp . . . . .	495
26.12airtsp/command/TravelSolutionParser.cpp File Reference . . . . .	496
26.124TravelSolutionParser.cpp . . . . .	497
26.12airtsp/command/TravelSolutionParser.hpp File Reference . . . . .	500
26.12TravelSolutionParser.hpp . . . . .	501
26.12airtsp/config/airtsp-paths.hpp.in File Reference . . . . .	502
26.128airtsp-paths.hpp.in . . . . .	503
26.12airtsp/factory/FacAIRTSPServiceContext.cpp File Reference . . . . .	504
26.13FacAIRTSPServiceContext.cpp . . . . .	505
26.13airtsp/factory/FacAIRTSPServiceContext.hpp File Reference . . . . .	506
26.13FacAIRTSPServiceContext.hpp . . . . .	507
26.13airtsp/factory/FacServiceAbstract.cpp File Reference . . . . .	508
26.13FacServiceAbstract.cpp . . . . .	509
26.13airtsp/factory/FacServiceAbstract.hpp File Reference . . . . .	510
26.13FacServiceAbstract.hpp . . . . .	511
26.13airtsp/service/AIRTSP_Service.cpp File Reference . . . . .	512
26.13AIRTSP_Service.cpp . . . . .	513
26.13airtsp/service/AIRTSP_ServiceContext.cpp File Reference . . . . .	520
26.14AIRTSP_ServiceContext.cpp . . . . .	521
26.14airtsp/service/AIRTSP_ServiceContext.hpp File Reference . . . . .	523
26.14AIRTSP_ServiceContext.hpp . . . . .	524
26.14airtsp/service/ServiceAbstract.cpp File Reference . . . . .	526
26.14ServiceAbstract.cpp . . . . .	527
26.14airtsp/service/ServiceAbstract.hpp File Reference . . . . .	528
26.145. Function Documentation . . . . .	528
26.14ServiceAbstract.hpp . . . . .	529
26.14doc/local/authors.doc File Reference . . . . .	530
26.148doc/local/codingrules.doc File Reference . . . . .	530
26.14doc/local/copyright.doc File Reference . . . . .	530
26.15doc/local/documentation.doc File Reference . . . . .	530
26.15doc/local/features.doc File Reference . . . . .	530
26.152doc/local/help_wanted.doc File Reference . . . . .	530
26.153doc/local/howto_release.doc File Reference . . . . .	530
26.154doc/local/index.doc File Reference . . . . .	530

<a href="#">26.158doc/local/installation.doc File Reference</a>	530
<a href="#">26.156doc/local/linking.doc File Reference</a>	530
<a href="#">26.157doc/local/test.doc File Reference</a>	530
<a href="#">26.158doc/local/users_guide.doc File Reference</a>	530
<a href="#">26.159doc/local/verification.doc File Reference</a>	530
<a href="#">26.160doc/tutorial/tutorial.doc File Reference</a>	530
<a href="#">26.161test/airtsp/AirlineScheduleTestSuite.cpp File Reference</a>	530
<a href="#">26.162AirlineScheduleTestSuite.cpp</a>	531
<a href="#">26.163test/airtsp/AirlineScheduleTestSuite.hpp File Reference</a>	536
<a href="#">26.163. Function Documentation</a>	536
<a href="#">26.164AirlineScheduleTestSuite.hpp</a>	537

## 1 AirTSP Documentation

### 1.1 Getting Started

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

### 1.2 AirTSP at SourceForge

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

## 1.3 AirTSP Development

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

## 1.4 External Libraries

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

## 1.5 Support AirTSP

## 1.6 About AirTSP

AirTSP is a C++ library of classes and functions modeling airline schedules, for instance allowing to retrieve all the flight-based travel solutions corresponding to a given pair of origin and destination points. AirTSP mainly targets simulation purposes. [N](#)

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

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

AirTSP 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 AirTSP 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 AirTSP.

## 2 Configuration helper for AirTSP programs

```
*/
#ifndef __AIRTSP_PATHS_HPP__
#define __AIRTSP_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@"

#endif // __AIRTSP_PATHS_HPP__

/*!
```

## 3 People

### 3.1 Project Admins

- Denis Arnaud <[denis\\_arnaud@users.sourceforge.net](mailto:denis_arnaud@users.sourceforge.net)> (N)
- Anh Quan Nguyen <[quannaus@users.sourceforge.net](mailto:quannaus@users.sourceforge.net)> (N)

### 3.2 Developers

- Anh Quan Nguyen <[quannaus@users.sourceforge.net](mailto:quannaus@users.sourceforge.net)> (N)
- Denis Arnaud <[denis\\_arnaud@users.sourceforge.net](mailto:denis_arnaud@users.sourceforge.net)> (N)
- Gabrielle Sabatier <[gsabatier@users.sourceforge.net](mailto:gsabatier@users.sourceforge.net)> (N)

### 3.3 Retired Developers

- Daniel Perez <[daniperez@users.sourceforge.net](mailto:daniperez@users.sourceforge.net)> (N)
- Mehdi Ayouni <[mehdi.ayouni@gmail.com](mailto:mehdi.ayouni@gmail.com)>
- Son Nguyen Kim <[snguyenkim@users.sourceforge.net](mailto:snguyenkim@users.sourceforge.net)>
- Alexandre Point <[apoint@users.sourceforge.net](mailto:apoint@users.sourceforge.net)>

### 3.4 Contributors

- Emmanuel Bastien <[ebastien@users.sourceforge.net](mailto:ebastien@users.sourceforge.net)> (N)
- Christophe Lacombe <[ddtoif@users.sourceforge.net](mailto:ddtoif@users.sourceforge.net)> (N)

## 3.5 Distribution Maintainers

- **Fedora/RedHat**: Denis Arnaud <denis\_arnaud@users.sourceforge.net> (N)
- **Debian**: Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

**Note:**

(N) - **Amadeus** employees.

## 4 Coding Rules

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

### 4.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

### 4.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

### 4.3 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`

### 4.4 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`

## 4.5 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.

## 5 Copyright and License

### 5.1 GNU LESSER GENERAL PUBLIC LICENSE

#### 5.1.1 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.]

### 5.2 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.

### **5.3 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

### **5.3.1 NO WARRANTY**

15. 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.

16. 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.



### 5.3.2 END OF TERMS AND CONDITIONS

## 5.4 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

## 6 Documentation Rules

### 6.1 General Rules

All classes in AirTSP 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 AirTSP 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
};

```

## 6.2 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.
 *
 * -----
 *
 * AirTSP - C++ Airline Travel Solution Provider Library
 *
 * Copyright (C) 2009-2010 (\see authors file for a list of contributors)
 *
 * \see copyright file for license information

```

```
*
* -----
*/
```

## 6.3 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);
```

## 7 Main features

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

### 7.1 Network generation

- Network/graph generation

### 7.2 Finding travel solutions

- Matching of travel solutions with user requests

### 7.3 Other features

- CSV input file parsing
- Memory handling

## 8 Make a Difference

**Do not ask what AirTSP can do for you. Ask what you can do for AirTSP.**

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

- Start using AirTSP
- Tell your friends about AirTSP 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 AirTSP discussion forums on SourceForge. If you know the answer to a question, help others to overcome their AirTSP 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 AirTSP to new platforms. If you manage to compile AirTSP on a new platform, then tell us how you did it.
- Send us your code. If you have a good AirTSP compatible code, which you can release under the LGPLv2.1, and you think it should be included in AirTSP, then send it to us.
- Become an AirTSP developer. Send us an e-mail and tell what you can do for AirTSP.

## 9 Make a new release

### 9.1 Introduction

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

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

### 9.2 Initialisation

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

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

### 9.3 Release branch maintenance

Switch to the release branch, on your local clone, and merge the latest updates from the trunk. Decide about the new version to be released.

```
cd ~/dev/sim/airtspgit
git checkout releases
git merge trunk
```

Update the version in the various build system files, replacing the old version numbers by the correct ones:

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

Update the version, add some news in the NEWS file, add a change-log in the ChangeLog file and in the RPM specification files:

```
vi NEWS
vi ChangeLog
vi airtsp.spec
```

## 9.4 Commit and publish the release branch

Commit the new release:

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

## 9.5 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/airtspgit
git checkout releases
rm -rf build && mkdir -p build
cd build
export INSTALL_BASEDIR=/home/user/dev/deliveries
export LIBSUFFIX_4_CMAKE="-DLIB_SUFFIX=64"
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_BASEDIR}/airtsp-0.5.0 \
      -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON \
      ${LIBSUFFIX_4_CMAKE} ..
make check && make dist
make install
```

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

## 9.6 Upload the HTML documentation to SourceForge

In order to update the Web site files, either:

- **synchronise them with rsync and SSH:** Upload the just generated HTML (and PDF) documentation onto the **SourceForge Web site**.

```
cd ~/dev/sim/airtspgit/build
git checkout releases
rsync -aiv ${INSTALL_BASEDIR}/airtsp-0.5.0/share/doc/airtsp-0.5.0/html/ \
  your_sf_user,airtsp@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](#).

## 9.7 Generate the RPM packages

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

```
cd ~/dev/sim/airtspgit/build
git checkout releases
make dist
```

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

```
cp ../airtsp.spec ~/dev/packages/SPECS \
  && cp airtsp-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba airtsp.spec
cd ~/dev/packages
rpmlint -i SPECS/airtsp.spec SRPMS/airtsp-0.5.0-1.fc16.src.rpm \
  RPMS/noarch/airtsp-* RPMS/i686/airtsp-*
```

## 9.8 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 [AirTSP's Git repository](#).

## 9.9 Create the binary package, including the documentation

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

```
cd ~/dev/sim/airtspgit/build
git checkout releases
make package
```

The output binary package will be named, for instance, `airtsp-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.

## 9.10 Upload the files to SourceForge

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

## 9.11 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](#).

## 9.12 Send an email on the announcement mailing-list

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

# 10 Installation

## 10.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [AirTSP 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](#)

## 10.2 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 airtsp-devel airtsp-doc
```

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

## 10.3 AirTSP Requirements

AirTSP 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'`)
- **SOCI** - 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 AirTSP.

## 10.4 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 `'CMakeLists.txt'` is used to create the `'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.

## 10.5 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.

## 10.6 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.

## 10.7 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.

## 10.8 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'`.

## 10.9 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
```

## 10.10 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'`.

## 10.11 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.

## 10.12 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
```

## 10.13 '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:

```
-- Requires Git without specifying any version
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airtsp-99.99.99 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:
-- 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: 6100bb1479e9c72f807a60067138dfelb71cbec7 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 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-0.35
-- 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 'airtsplib' to CXX
-- Test 'AirlineScheduleTestSuite' to be built with 'AirlineScheduleTestSuite.cpp'
--
-- =====
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : airtsp
-- PACKAGE_PRETTY_NAME ..... : AirTSP
-- PACKAGE ..... : airtsp
-- PACKAGE_NAME ..... : AIRTSP
-- PACKAGE_BRIEF ..... : C++ Simulated Airline Schedule Manager Library
-- PACKAGE_VERSION ..... : 99.99.99
-- GENERIC_LIB_VERSION ..... : 99.99.99
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : airtsp
-- Libraries to build/install ..... : airtsplib
-- Binaries to build/install ..... : airtsp
-- Modules to test ..... : airtsp
-- Binaries to test ..... : AirlineScheduleTestSuite.tst
--
-- * Module ..... : airtsp
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :

```

```

-- + Libraries to build/install . : airtsp-lib
-- + Executables to build/install : airtsp
-- + Tests to perform ..... : AirlineScheduleTestSuiteTest
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -Wall -Werror
-- * BUILD_FLAGS ..... :
-- * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/user/dev/sim/airtsp/airtspgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/user/dev/deliveries/airtsp-99.99.99
--
-- * 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/user/dev/deliveries/airtsp-99.99.99/lib64
-- INSTALL_BIN_DIR ..... : /home/user/dev/deliveries/airtsp-99.99.99/bin
-- INSTALL_INCLUDE_DIR ..... : /home/user/dev/deliveries/airtsp-99.99.99/include
-- INSTALL_DATA_DIR ..... : /home/user/dev/deliveries/airtsp-99.99.99/share
-- INSTALL_SAMPLE_DIR ..... : /home/user/dev/deliveries/airtsp-99.99.99/share/airtsp/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 ..... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/airtsp/airtspgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/airtsp/airtspgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : airtsp-99.99.99
--
-- -----
-- --- 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;u
-- - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_
--
-- * 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/user/dev/deliveries/stdair-0.38.0/bin
-- - STDAIR_EXECUTABLES ..... : stdair
-- - STDAIR_LIBRARY_DIRS ..... : /home/user/dev/deliveries/stdair-0.38.0/lib64
-- - STDAIR_LIBRARIES ..... : stdairlib;stdairuicllib
-- - STDAIR_INCLUDE_DIRS ..... : /home/user/dev/deliveries/stdair-0.38.0/include
-- - STDAIR_SAMPLE_DIR ..... : /home/user/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/user/dev/sim/airtsp/airtspgithub/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_airtsp
[ 96%] Built target airtsplib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airtsptst
Test project /home/dan/dev/sim/airtsp/airtspgithub/build/test/airtsp
  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_airtsptst
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/airtspgit
rm -rf build && mkdir build
cd build
```

to remove everything.



## 11 Linking with AirTSP

### 11.1 Table of Contents

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

### 11.2 Introduction

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

### 11.3 Dependencies

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

#### 11.3.1 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'`).

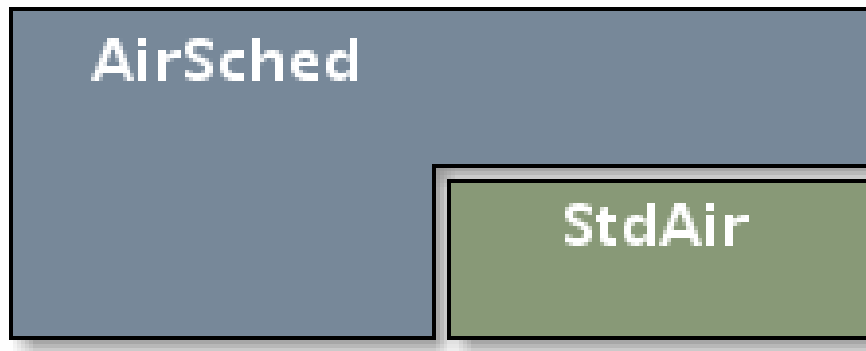


Figure 1: AirTSP Dependencies

## 11.4 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 AirTSP based program 'my\_prog.cpp', you should use the following command:

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

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

## 11.5 Using the airtsp-config script

AirTSP provides a shell script called 'airtsp-config', which is installed by default in '\$prefix/bin' ('/usr/local/bin') directory. It can be used to simplify compilation and linking of AirTSP 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++ `airtsp-config --cflags` -o my_prog_opt my_prog.cpp `airtsp-config --libs`
```

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

```
airtsp-config --help
```

If the `'airtsp-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
```

## 11.6 M4 macro for the GNU Autotools

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

- `'AirTSP_VERSION'` (e.g., defined to 0.23.0)
- `'AirTSP_CFLAGS'` (e.g., defined to `'-I${prefix}/include'`)
- `'AirTSP_LIBS'` (e.g., defined to `'-L${prefix}/lib -lairtsp'`)

## 11.7 Using AirTSP with dynamic linking

When using static linking some of the library routines in AirTSP 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 AirTSP library file during your program execution. If you install the AirTSP 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=<AirTSP installation prefix>/lib:$LD_LIBRARY_PATH
```

# 12 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.

## 12.1 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'`.

## 12.2 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.

## 12.3 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.

# 13 Users Guide

## 13.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
  - [Get the AirTSP library](#)
  - [Build the AirTSP project](#)
  - [Build and Run the Tests](#)
  - [Install the AirTSP Project \(Binaries, Documentation\)](#)
- [Input file of AirTSP 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](#)

## 13.2 Introduction

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

## 13.3 Get Started

### 13.3.1 Get the AirTSP library

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

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

### 13.3.2 Build the AirTSP project

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

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

### 13.3.3 Build and Run the Tests

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

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

As a result, you should obtain a similar report:

```
[ 0%] Built target hdr_cfg_airtsp
[ 96%] Built target airtspplib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airsptst
Test project /home/dan/dev/sim/airtsp/airtspgithub/build/test/airtsp
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_airsptst
Scanning dependencies of target check
[100%] Built target check
```

### 13.3.4 Install the AirTSP Project (Binaries, Documentation)

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

```
cd ~/dev/sim/airtspgit
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/airtsp-0.5.0
```

To generate the AirTSP project documentation, the commands are:

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

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

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

## 13.4 Input file of AirTSP Project

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

```
\textcolor{comment}{// Flights:   AirlineCode; FlightNumber; Date-Range; ; DOW; Legs; Segments;}
\textcolor{comment}{// Legs:      BoardPoint; OffPoint; BoardTime; ArrivalDateOffSet; ArrivalTime;}
\textcolor{comment}{//              ElapsedTime; LegCabins;}
\textcolor{comment}{// LegCabins: CabinCode; Capacity;}
\textcolor{comment}{// Segments: Specific; }
BA; 9; 2007-04-20; 2007-06-30; 0000011; LHR; BKK; 22:00; 15:15 / +1; 11:15; F; 5;
    J; 12; W; 20; Y; 300; BKK; SYD; 18:10 / +1; 06:05 / +2; 08:55; F; 5; J; 12; W; 2
    0; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKMLSQ; 1; YBHKM
    LSQ;
BA; 9; 2007-04-20; 2007-06-30; 1111100; LHR; BKK; 22:00; 15:15 / +1; 11:15; F; 5;
    J; 12; W; 20; Y; 300; BKK; SYD; 18:10 / +1; 06:05 / +2; 08:55; F; 5; J; 12; W; 2
    0; Y; 300; 1; LHR; BKK; F; FA; J; JC DI; W; WT; Y; YBHKMLSQ; BKK; SYD; F; FA; J; J
    C DI; W; WT; Y; YBHKMLSQ; LHR; SYD; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT;
    Y; YBHKMLSQ; 1; YBHKMLSQ;
BA; 117; 2007-04-20; 2007-06-30; 1111111; LHR; JFK; 08:20; 11:00; 07:40; F; 5; J;
    12; W; 20; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKM; 1;
    YBHKM;
BA; 175; 2007-04-20; 2007-06-30; 1111111; LHR; JFK; 10:55; 13:35; 07:40; F; 5; J;
    12; W; 20; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKMRL;
    1; YBHKMRL;
BA; 179; 2007-04-20; 2007-06-30; 1111111; LHR; JFK; 18:05; 20:45; 07:40; F; 5; J;
    12; W; 20; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKMRVNE
    LSQO; 1; YBHKMRVNELSQO;
BA; 207; 2007-04-20; 2007-06-30; 1111111; LHR; MIA; 09:40; 14:25; 09:45; F; 5; J;
    12; W; 20; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKMRVNE
    LSQO; 1; YBHKMRVNELSQO;
BA; 279; 2007-04-20; 2007-06-30; 1111111; LHR; LAX; 10:05; 13:10; 11:05; F; 5; J;
    12; W; 20; Y; 300; 0; F; FA; 1; FA; J; JC DI; 1; JC DI; W; WT; 1; WT; Y; YBHKMRVNE
    LSQO; 1; YBHKMRVNELSQO;
```

Each line, beyond the header, represents a schedule entry, i.e., the specification of a given flight-period (see [AIRTSP::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`.

## 13.5 The schedule BOM Tree

The schedule-related Business Object Model (BOM) tree is a structure allowing to store all the `AIRTSP::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 AirTSP 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.

### 13.5.1 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 `AIRTSP::AIRTSP_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 `AIRTSP::AIRTSP_Service::buildSampleBom()` method:

```
\textcolor{keywordtype}{void} buildSampleBom();
```

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

```
\textcolor{keywordtype}{void} parseAndLoad (\textcolor{keyword}{const} stdair::ScheduleFilePath&);
```

### 13.5.2 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::airtsp.cpp` program:

When the default BOM tree is used (`-b/--builtin` option of the main program `airtsp.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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm
    Space, 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, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-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-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, El
    apsed, 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, Comm
    Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm
Space, 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-31
9, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, El
      apsed, 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, Comm
      Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm
      Space, 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, El
    apsed, 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, Comm
    Space, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
  Space, 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, El
  apsed, 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, Comm
  Space, 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, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, El
      apsed, Distance, Capacity,
SQ11 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 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,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 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,
SQ11 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,
SQ11 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: SQ11, 2010-Feb-10
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL1 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,
SQL1 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, 0, 0, 0, 0, 300, 0,
SQL1 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, El
    apsed, 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, Comm
    Space, 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, El
    apsed, 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, Comm
    Space, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm

```

```

Space, 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, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, OfferedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
Space, 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, 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, El
apsed, 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, Comm
    Space, 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, El
    apsed, 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, Comm
    Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El

```

```

        apsed, 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, Comm
        Space, 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, 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, El
        apsed, 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, Comm
        Space, 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, 5.20
        268e-90, 0, 1.31346e-47, 1.05119e-153, 2.78986e+179, 0, 200, 9, 3.66962e-62, 1.08
        54e-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, El  
apsed, 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, Comm  
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,  
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 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,  
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,  
SQ12 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,  
SQ12 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,  
SQ12 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: SQ12, 2010-Jan-17

\*\*\*\*\*  
\*\*\*\*\*

Leg-Dates:

-----

Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El  
apsed, Distance, Capacity,  
SQ12 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, Comm  
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,  
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 200, 200, 0, 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,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 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,
SQ12 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,
SQ12 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: SQ12, 2010-Jan-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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-Jan-20, SIN-HND 2010-Jan-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Jan-21
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQL2 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 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,
SQL2 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Jan-22
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQL2 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, Comm
    Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 2010-Jan-22, SIN-HND 2010-Jan-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,
SQL12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 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,
SQL12 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,
SQL12 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: SQL12, 2010-Jan-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
    apsed, Distance, Capacity,
SQL12 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, Comm
    Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL12 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,
SQL12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQL12 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,
SQL12 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,
SQL12 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, El
apsed, 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, Comm
Space, 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,
SQ12 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,
SQ12 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: SQ12, 2010-Jan-25
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQ12 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, Comm
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, 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-25, SIN-HND 2010-Jan-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 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,
SQ12 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,
SQ12 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: SQ12, 2010-Jan-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 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,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 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,
SQ12 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,
SQ12 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, El
      apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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-28, SIN-HND 2010-Jan-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-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-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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El  
 apsed, 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, Comm  
 Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, El
      apsed, 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, Comm
      Space, 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, 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, El
      apsed, 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,

```



```
SQL2 2010-Feb-04, SIN-HND 2010-Feb-04, 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-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Feb-05
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
  apsed, Distance, Capacity,
SQL2 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, OfferedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
  Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 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,
SQL2 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
  Space, 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, 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, El
  apsed, 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, Comm
  Space, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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-12, SIN-HND 2010-Feb-12, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, 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-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, El
      apsed, 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, Comm
      Space, 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, 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, El
      apsed, Distance, Capacity,

```

```
SQL2 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, Comm
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Feb-14, SIN-HND 2010-Feb-14, 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-14, SIN-HND 2010-Feb-14, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Feb-15
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQL2 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, Comm
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Feb-15, SIN-HND 2010-Feb-15, 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-15, SIN-HND 2010-Feb-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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, 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, El
    apsed, 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, Comm
    Space, 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, 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, El
    apsed, 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, Comm
    Space, 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, El
      apsed, 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, Comm
      Space, 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, 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, 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, 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, El
      apsed, 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, Comm
      Space, 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, El
      apsed, 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, Comm

```

```

Space, 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, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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, 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, El
apsed, 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, Comm
Space, 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, 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,
SQ12 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,
SQ12 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,
SQ12 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: SQ12, 2010-Feb-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQ12 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, OfferedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-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,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 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,
SQ12 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,
SQ12 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: SQ12, 2010-Feb-27
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQ12 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, Comm
    Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 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,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 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, El
    apsed, 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, Comm
    Space, 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, 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,
*****

```

## 13.6 Exploring the Predefined BOM Tree

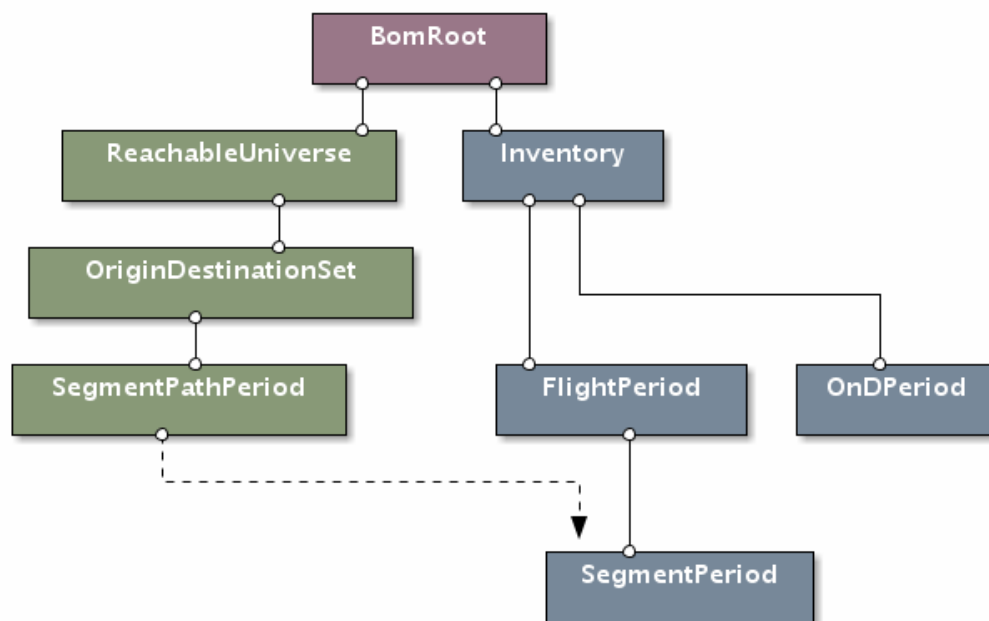


Figure 2: AirTSP BOM tree

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

### 13.6.1 Airline Network BOM Tree

- `AIRTSP::ReachableUniverse`
- `AIRTSP::OriginDestinationSet`
- `AIRTSP::SegmentPathPeriod`

### 13.6.2 Airline Schedule BOM Tree

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



- `stdair::OnDPeriod`

## 13.7 Extending the BOM Tree

## 13.8 The travel solution calculation procedure

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

# 14 Supported Systems

## 14.1 Table of Contents

- [Introduction](#)
- [AirTSP 0.2.x](#)
  - [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 AirTSP 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 AirTSP External](#)
    - \* [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
  - [Unix Systems](#)
    - \* [SunOS 5.9 with AirTSP External](#)
- [AirTSP 3.9.1](#)
- [AirTSP 3.9.0](#)
- [AirTSP 3.8.1](#)

## 14.2 Introduction

This page is intended to provide a list of AirTSP supported systems, i.e. the systems on which configuration, installation and testing process of the AirTSP 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 AirTSP library on a system not mentioned below, please let us know, so we could update this database.

## 14.3 AirTSP 0.2.x

### 14.3.1 Linux Systems

#### 14.3.1.1 Fedora Core 4 with ATLAS

- **Platform:** Intel Pentium 4
- **Operating System:** Fedora Core 4 (x86)
- **Compiler:** g++ (GCC) 4.0.2 20051125
- **AirTSP release:** 0.2.0
- **External Libraries:** From FC4 distribution:
  - `fftw3.i386-3.0.1-3`
  - `fftw3-devel.i386-3.0.1-3`
  - `atlas-sse2.i386-3.6.0-8.fc4`
  - `atlas-sse2-devel.i386-3.6.0-8.fc4`
  - `blas.i386-3.0-35.fc4`
  - `lapack.i386-3.0-35.fc4`
- **Tests Status:** All tests PASSED
- **Comments:** AirTSP configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```
- **Date:** March 7, 2006
- **Tester:** Tony Ottosson

#### 14.3.1.2 Gentoo Linux with ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s):** g++ (GCC) 3.4.5
- **AirTSP release:** 0.2.1
- **External Libraries:** Compiled and installed from portage tree:
  - `sci-libs/acml-3.0.0`
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

AirTSP configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```

- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.1.3 Gentoo Linux with ATLAS

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **AirTSP release:** 0.2.1
- **External Libraries:** Compiled and installed from portage tree:
  - sci-libs/fftw-3.1
  - sci-libs/blas-atlas-3.6.0-r1
  - sci-libs/lapack-atlas-3.6.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

AirTSP configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.1.4 Gentoo Linux with MKL

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler:** g++ (GCC) 3.4.5
- **AirTSP release:** 0.2.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED

- **Comments:** AirTSP configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date:** February 28, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **AirTSP release:** 0.2.1
- **External Libraries:** Compiled and installed from portage tree:
  - sci-libs/fftw-3.1
  - sci-libs/blas-reference-19940131-r2
  - sci-libs/cblas-reference-20030223
  - sci-libs/lapack-reference-3.0-r2
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

AirTSP configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.1.6 Red Hat Enterprise Linux with AirTSP External

- **Platform:** Intel Pentium 4
- **Operating System:** Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler:** g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **AirTSP release:** 0.2.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirTSP External 2.1.1 package
- **Tests Status:** All tests PASSED
- **Date:** March 7, 2006
- **Tester:** Erik G. Larsson

#### 14.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86\_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **AirTSP release:** 0.2.0
- **External Libraries:** BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:

```
- blas-3.0-926
- lapack-3.0-926
- fftw3-3.0.1-114
- fftw3-threads-3.0.1-114
- fftw3-devel-3.0.1-114
```

- **Tests Status:** All tests PASSED
- **Comments:** AirTSP configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```

- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.1.8 SUSE Linux 10.0 with MKL

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86\_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **AirTSP release:** 0.2.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: /opt/intel/mkl/8.0.1

- **Tests Status:** All tests PASSED
- **Comments:** AirTSP configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

### 14.3.2 Windows Systems

#### 14.3.2.1 Microsoft Windows XP with Cygwin

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **AirTSP release:** 0.2.1
- **External Libraries:** Installed from Cygwin's repository:
  - fftw-3.0.1-2
  - fftw-dev-3.0.1-1
  - lapack-3.0-4
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirTSP configured with:

```
% ./configure
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **AirTSP release:** 0.2.1
- **External Libraries:** Installed from Cygwin's repository:
  - fftw-3.0.1-2
  - fftw-dev-3.0.1-1ATLAS BLAS and LAPACK libraries from AirTSP External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirTSP configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

### 14.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **AirTSP release:** 0.2.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirTSP configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

### 14.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **AirTSP release:** 0.2.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirTSP configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.2.5 Microsoft Windows XP with MinGW, MSYS and AirTSP External

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **AirTSP release:** 0.2.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirTSP External 2.2.0 package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirTSP configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 14.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2
- **Compiler(s):** Microsoft Visual C++ 2005 .NET
- **AirTSP release:** 0.2.5
- **External Libraries:** Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "C:\Program Files\Intel\MKL\8.1"
- **Tests Status:** Not fully tested. Some AirTSP based programs compiled and run with success.
- **Comments:** Only static library can be built. AirTSP built by opening the "win32\airtsp.vcproj" project file in MSVC++ and executing "Build -> Build Solution" command from menu.
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

### 14.3.3 Unix Systems

#### 14.3.3.1 SunOS 5.9 with AirTSP External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic\_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5



- **AirTSP release:** 0.2.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirTSP External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"  
% ./configure
```

- **Tests Status:** All tests PASSED
- **Comments:** AirTSP configured with:

```
% export LDFLAGS="-L/usr/local/lib"  
% export CPPFLAGS="-I/usr/local/include"  
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"  
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

## 15 AirTSP Supported Systems (Previous Releases)

### 15.1 AirTSP 3.9.1

### 15.2 AirTSP 3.9.0

### 15.3 AirTSP 3.8.1

## 16 Tutorials

### 16.1 Table of Contents

- [Preparing the AirTSP 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](#)

### 16.2 Preparing the AirTSP Project for Development

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

## 16.3 Your first networkBuilder

### 16.3.1 Summary of the different steps

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

First, we instantiate the `AIRTSP_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):

```
stdair::TravelSolutionList lTravelSolutionList;  
airtspService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
```

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

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

### 16.3.2 Result of the Batch Program

When the `AirTSP.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.

## 16.4 Network building with an input file

### 16.4.1 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 :

```
\textcolor{comment}{// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments}
\textcolor{comment}{// Segment: AirlineCode; Class; }
1; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ???; ?; ??; ?; ?; ?; ?; ?;
    ???; BA; ?;
```

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:

```
\textcolor{comment}{// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments}
\textcolor{comment}{// Segment: AirlineCode; Class; }
1; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; IN; 0; ?; ?; ?; 0;
    ???; BA; ?;
2; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; IF; 0; ?; ?; ?; 0;
    ???; BA; ?;
3; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; DN; 0; ?; ?; ?; 0;
    ???; BA; ?;
4; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; DF; 0; ?; ?; ?; 0;
    ???; BA; ?;
5; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; IN; 0; ?; ?; ?; 0;
    ???; BA; ?;
6; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; IF; 0; ?; ?; ?; 0;
    ???; BA; ?;
```

```

7; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; DN; 0; ?; ?; ?; 0;
   ????; BA; ?;
8; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; DF; 0; ?; ?; ?; 0;
   ????; BA; ?;
9; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; IN; 0; ?; ?; ?; 0;
   ????; BA; ?;
10; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; IF; 0; ?; ?; ?; 0;
   ; ????; BA; ?;
11; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; DN; 0; ?; ?; ?; 0;
   ; ????; BA; ?;
12; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; DF; 0; ?; ?; ?; 0;
   ; ????; BA; ?;

```

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:

```

\textcolor{comment}{// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments}
\textcolor{comment}{// Segment: AirlineCode; Class; }
1; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IN; 0; T; T; T; 0;
   50; BA; Y;
2; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IF; 0; T; T; T; 0;
   150; BA; Y;
3; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DN; 0; T; T; T; 0;
   250; BA; Y;
4; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DF; 0; T; T; T; 0;
   350; BA; Y;
5; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IN; 0; T; T; T; 0;
   450; BA; Y;
6; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IF; 0; T; T; T; 0;
   550; BA; Y;
7; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DN; 0; T; T; T; 0;
   650; BA; Y;
8; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DF; 0; T; T; T; 0;
   750; BA; Y;
9; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IN; 0; T; T; T; 0;
   850; BA; Y;
10; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IF; 0; T; T; T; 0;
   ; 950; BA; Y;
11; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DN; 0; T; T; T; 0;
   ; 1050; BA; Y;
12; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DF; 0; T; T; T; 0;
   ; 1150; BA; Y;
13; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IN; 0; T; T; T; 0;
   ; 90; BA; Y;
14; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IF; 0; T; T; T; 0;
   ; 190; BA; Y;
15; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DN; 0; T; T; T; 0;
   ; 290; BA; Y;
16; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DF; 0; T; T; T; 0;
   ; 390; BA; Y;
17; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IN; 0; T; T; T; 0;
   ; 490; BA; Y;
18; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IF; 0; T; T; T; 0;
   ; 590; BA; Y;
19; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DN; 0; T; T; T; 0;
   ; 690; BA; Y;

```

```

20; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DF; 0; T; T; T; 0
    ; 790; BA; Y;
21; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IN; 0; T; T; T; 0
    ; 890; BA; Y;
22; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IF; 0; T; T; T; 0
    ; 990; BA; Y;
23; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DN; 0; T; T; T; 0
    ; 1090; BA; Y;
24; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DF; 0; T; T; T; 0
    ; 1190; BA; Y;

```

#### 16.4.2 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 :

#### 16.4.3 Result of the Batch Program

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

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

the last lines of the log output should look like:

```

[D]~/AirTSPgit/AirTSP/batches/AirTSP.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.

## 17 Command-Line Test to Demonstrate How To Test the AirTSP 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>
// Boost Date-Time
#include <boost/date_time/gregorian/gregorian.hpp>
// StdAir
#include <stdair/basic/BasFileMgr.hpp>
#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>
// AirTSP
#include <airtsp/AIRTSP_Types.hpp>
#include <airtsp/AIRTSP_Service.hpp>
#include <airtsp/config/airtsp-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("AirlineScheduleTestSuite_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() {
    }
};

// ////////////////////////////////////////
const unsigned int testScheduleHelper (const unsigned short iTestFlag,
                                       const stdair::Filename_T& iScheduleInputFilename,
                                       const stdair::Filename_T& iODInputFilename,
                                       const bool isBuiltin,
                                       const bool isWithOnD) {

    // Output log File
    std::ostream oStr;
    oStr << "AirlineScheduleTestSuite_" << 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();

    // Instantiate the AirTSP service
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
    AIRTSP::AIRTSP_Service airtspService (lLogParams);

    stdair::AirportCode_T lOrigin;
    stdair::AirportCode_T lDestination;
    stdair::AirportCode_T lPOS;
    stdair::Date_T lPreferredDepartureDate;
    stdair::Date_T lRequestDate;

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

        // Build the default sample BOM tree (filled with schedules)
        airtspService.buildSampleBom();

        lOrigin = "SIN";
        lDestination = "BKK";
        lPOS = "SIN";
        lPreferredDepartureDate = boost::gregorian::from_string ("2010/02/08");
        lRequestDate = boost::gregorian::from_string ("2010/01/21");
    }
}

```

```

} else {

    if (isWithOnD == false) {

        // Build the BOM tree from parsing input files
        const stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
        airtspService.parseAndLoad (lScheduleFilePath);

        lOrigin = "NCE";
        lDestination = "BKK";
        lPOS = "NCE";
        lPreferredDepartureDate = boost::gregorian::from_string ("2007/04/21");
        lRequestDate = boost::gregorian::from_string ("2007/03/21");

    } else {

        // Build the BOM tree from parsing input files
        const stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
        const stdair::ODFilePath lODFilePath (iODInputFilename);
        airtspService.parseAndLoad (lScheduleFilePath,
                                    lODFilePath);

        lOrigin = "SIN";
        lDestination = "BKK";
        lPOS = "SIN";
        lPreferredDepartureDate = boost::gregorian::from_string ("2012/06/04");
        lRequestDate = boost::gregorian::from_string ("2012/01/01");
    }
}

// Create a booking request structure
const stdair::Duration_T lRequestTime (boost::posix_time::hours(8));
const stdair::DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
const stdair::CabinCode_T lPreferredCabin ("Bus");
const stdair::PartySize_T lPartySize (3);
const stdair::ChannelLabel_T lChannel ("DF");
const stdair::TripType_T lTripType ("RO");
const stdair::DayDuration_T lStayDuration (5);
const stdair::FrequentFlyer_T lFrequentFlyerType ("NONE");
const stdair::Duration_T lPreferredDepartureTime (boost::posix_time::hours(10))
;
const stdair::WTP_T lWTP (2000.0);
const stdair::PriceValue_T lValueOfTime (20.0);
const stdair::ChangeFees_T lChangeFees (true);
const stdair::Disutility_T lChangeFeeDisutility (50);
const stdair::NonRefundable_T lNonRefundable (true);
const stdair::Disutility_T lNonRefundableDisutility (50);

const stdair::BookingRequestStruct lBookingRequest (lOrigin, lDestination,
                                                    lPOS,
                                                    lPreferredDepartureDate,
                                                    lRequestDateTime,
                                                    lPreferredCabin,
                                                    lPartySize, lChannel,
                                                    lTripType, lStayDuration,
                                                    lFrequentFlyerType,
                                                    lPreferredDepartureTime,
                                                    lWTP, lValueOfTime,
                                                    lChangeFees,
                                                    lChangeFeeDisutility,
                                                    lNonRefundable,
                                                    lNonRefundableDisutility);

// Build the segment path list
stdair::TravelSolutionList_T lTravelSolutionList;

```

```

airtspService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
const unsigned int lNbOfTravelSolutions = lTravelSolutionList.size();

STDAIR_LOG_DEBUG ("The number of travel solutions for the booking request '"
    << lBookingRequest.describe() << "' is equal to "
    << lNbOfTravelSolutions << ".");

// Close the Log outputFile
logOutputFile.close();

return lNbOfTravelSolutions;
}

// ////////////////////////////////// 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 (airtsp_simple_build) {

    // Input file name
    const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
        "/schedule03.csv");

    // State whether the BOM tree should be built-in or parsed from input files
    const bool isBuiltin = false;
    const bool isWithOnD = false;

    // Try to build a travel solution list
    unsigned int lNbOfTravelSolutions = 0;
    BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
        testScheduleHelper (0, lScheduleInputFilename, " ",
            isBuiltin, isWithOnD));

    // Check the size of the travel solution list
    const unsigned int lExpectedNbOfTravelSolutions = 4;
    BOOST_CHECK_MESSAGE (lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
        "The number of travel solutions is "
        << lNbOfTravelSolutions << ", but it should be equal to "
        << lExpectedNbOfTravelSolutions);

}

BOOST_AUTO_TEST_CASE (airtsp_default_bom_tree_simple_build) {

    // State whether the BOM tree should be built-in or parsed from input files
    const bool isBuiltin = true;
    const bool isWithOnD = false;

    // Try to build a travel solution list
    unsigned int lNbOfTravelSolutions = 0;
    BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
        testScheduleHelper (1, " ", " ", isBuiltin, isWithOnD));

    // Check the size of the travel solution list
    const unsigned int lExpectedNbOfTravelSolutions = 1;
    BOOST_CHECK_MESSAGE (lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
        "The number of travel solutions is "
        << lNbOfTravelSolutions << ", but it should be equal to "
        << lExpectedNbOfTravelSolutions);

}

```



```

BOOST_AUTO_TEST_CASE (airtsp_OnD_input_file) {

    // Input file names
    const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                    "/rds01/schedule05.csv");
    const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                              "/ond01.csv");

    // State whether the BOM tree should be built-in or parsed from input files
    const bool isBuiltin = false;
    const bool isWithOnD = true;

    // Try to build a travel solution list
    unsigned int lNbOfTravelSolutions = 0;
    BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
                          testScheduleHelper (2, lScheduleInputFilename,
                                              lODInputFilename,
                                              isBuiltin, isWithOnD));

    // Check the size of the travel solution list
    const unsigned int lExpectedNbOfTravelSolutions = 1;
    BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
                      "The number of travel solutions is "
                      "<< lNbOfTravelSolutions << ", but it should be equal to "
                      "<< lExpectedNbOfTravelSolutions);
}

BOOST_AUTO_TEST_CASE (airtsp_missing_OnD_input_file) {

    // Input file names
    const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                    "/schedule03.csv");
    const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                              "/missingFiles.csv");

    // State whether the BOM tree should be built-in or parsed from input files
    const bool isBuiltin = false;
    const bool isWithOnD = true;

    // Try to build a travel solution list
    BOOST_CHECK_THROW (testScheduleHelper (3, lScheduleInputFilename,
                                          lODInputFilename,
                                          isBuiltin, isWithOnD),
                      AIRTSP::OnDInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airtsp_missing_schedule_input_file) {

    // Input file name
    const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                    "/missingFiles.csv");

    // State whether the BOM tree should be built-in or parsed from input files
    const bool isBuiltin = false;
    const bool isWithOnD = false;

    // Try to build a travel solution list
    BOOST_CHECK_THROW (testScheduleHelper (4, lScheduleInputFilename, " ",
                                          isBuiltin, isWithOnD),
                      AIRTSP::ScheduleInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airtsp_segment_date_not_found) {

    // Input file name

```

```

const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                "/scheduleError03.csv");

// State whether the BOM tree should be built-in or parsed from input files
const bool isBuiltin = false;
const bool isWithOnD = false;

// Try to build a travel solution list
BOOST_CHECK_THROW (testScheduleHelper (5, lScheduleInputFilename,
                                     " ",
                                     isBuiltin, isWithOnD),
                  AIRTSP::SegmentDateNotFoundException);

}

// End the test suite
BOOST_AUTO_TEST_SUITE_END ()

/*!

```

## 18 Directory Hierarchy

### 18.1 Directories

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

<b>airtsp</b>	<b><a href="#">119</a></b>
<b>basic</b>	<b><a href="#">119</a></b>
<b>batches</b>	<b><a href="#">119</a></b>
<b>bom</b>	<b><a href="#">120</a></b>
<b>command</b>	<b><a href="#">120</a></b>
<b>config</b>	<b><a href="#">121</a></b>
<b>factory</b>	<b><a href="#">121</a></b>
<b>service</b>	<b><a href="#">121</a></b>
<b>test</b>	<b><a href="#">121</a></b>
<b>airtsp</b>	<b><a href="#">119</a></b>

## 19 Namespace Index

### 19.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b><a href="#">airtsp</a></b>	<b><a href="#">122</a></b>
-------------------------------	----------------------------

<a href="#">AIRTSP</a>	124
<a href="#">AIRTSP::OnDParserHelper</a>	132
<a href="#">AIRTSP::ScheduleParserHelper</a>	135
<a href="#">boost</a> (Forward declarations )	139
<a href="#">boost::serialization</a>	139
<a href="#">stdair</a> (Forward declarations )	139

## 20 Class Index

### 20.1 Class Hierarchy

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

<a href="#">airtsp::Airline_T</a>	139
<a href="#">AIRTSP::AIRTSP_Service</a>	142
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 >	
<a href="#">BomAbstract</a>	147
<a href="#">AIRTSP::OriginDestinationSet</a>	199
<a href="#">AIRTSP::ReachableUniverse</a>	213
<a href="#">AIRTSP::SegmentPathPeriod</a>	228
<a href="#">AIRTSP::BomDisplay</a>	147

<b>CmdAbstract</b>	<b>148</b>
AIRTSP::FlightPeriodFileParser	174
AIRTSP::InventoryGenerator	184
AIRTSP::OnDParser	190
AIRTSP::OnDPeriodFileParser	192
AIRTSP::OnDPeriodGenerator	193
AIRTSP::ScheduleParser	220
AIRTSP::SegmentPathGenerator	227
AIRTSP::SegmentPathProvider	239
AIRTSP::Simulator	244
AIRTSP::TravelSolutionParser	300
airtsp::Date_T	149
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >	151
airtsp::SearchStringParser::definition< ScannerT >	158
AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >	162
AIRTSP::FacServiceAbstract	170
FacServiceAbstract	171
AIRTSP::FacAIRTSPServiceContext	168
FileNotFoundException	173
AIRTSP::OnDInputFileNotFoundException	190
AIRTSP::ScheduleInputFileNotFoundException	220
AIRTSP::FlagSaver	173
grammar	184
AIRTSP::OnDParserHelper::OnDParser	191
AIRTSP::ScheduleParserHelper::FlightPeriodParser	175
airtsp::SearchStringParser	223
KeyAbstract	185
AIRTSP::OriginDestinationSetKey	204
AIRTSP::ReachableUniverseKey	217

AIRTSP::SegmentPathPeriodKey	234
ParserException	206
AIRTSP::SegmentDateNotFoundException	226
AIRTSP::OnDParserHelper::ParserSemanticAction	206
AIRTSP::OnDParserHelper::doEndOnD	166
AIRTSP::OnDParserHelper::storeAirlineCode	254
AIRTSP::OnDParserHelper::storeClassCode	260
AIRTSP::OnDParserHelper::storeDateRangeEnd	264
AIRTSP::OnDParserHelper::storeDateRangeStart	266
AIRTSP::OnDParserHelper::storeDestination	268
AIRTSP::OnDParserHelper::storeEndRangeTime	273
AIRTSP::OnDParserHelper::storeOrigin	290
AIRTSP::OnDParserHelper::storeStartRangeTime	298
AIRTSP::ScheduleParserHelper::ParserSemanticAction	208
AIRTSP::ScheduleParserHelper::doEndFlight	165
AIRTSP::ScheduleParserHelper::storeAirlineCode	255
AIRTSP::ScheduleParserHelper::storeBoardingTime	257
AIRTSP::ScheduleParserHelper::storeCapacity	258
AIRTSP::ScheduleParserHelper::storeClasses	261
AIRTSP::ScheduleParserHelper::storeDateRangeEnd	263
AIRTSP::ScheduleParserHelper::storeDateRangeStart	267
AIRTSP::ScheduleParserHelper::storeDow	270
AIRTSP::ScheduleParserHelper::storeElapsedTime	271
AIRTSP::ScheduleParserHelper::storeFamilyCode	274
AIRTSP::ScheduleParserHelper::storeFClasses	275
AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey	277
AIRTSP::ScheduleParserHelper::storeFlightNumber	278
AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey	280
AIRTSP::ScheduleParserHelper::storeLegBoardingPoint	281

AIRTSP::ScheduleParserHelper::storeLegCabinCode	283
AIRTSP::ScheduleParserHelper::storeLegOffPoint	284
AIRTSP::ScheduleParserHelper::storeOffTime	286
AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode	287
AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber	289
AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint	292
AIRTSP::ScheduleParserHelper::storeSegmentCabinCode	293
AIRTSP::ScheduleParserHelper::storeSegmentOffPoint	295
AIRTSP::ScheduleParserHelper::storeSegmentSpecificity	296
airtsp::Passenger_T	209
airtsp::Place_T	211
airtsp::SearchString_T	221
AIRTSP::SegmentPeriodHelper	240
ServiceAbstract	243
AIRTSP::AIRTSP_ServiceContext	146
AIRTSP::ServiceAbstract	243
airtsp::store_adult_passenger_type	245
airtsp::store_airline_code	246
airtsp::store_airline_name	247
airtsp::store_airline_sign	248
airtsp::store_child_passenger_type	249
airtsp::store_date	250
airtsp::store_passenger_number	251
airtsp::store_pet_passenger_type	252
airtsp::store_place_element	253
StructAbstract	299
AIRTSP::FareFamilyStruct	171
AIRTSP::FlightPeriodStruct	176
AIRTSP::LegCabinStruct	185

<b>AIRTSP::LegStruct</b>	<b>187</b>
<b>AIRTSP::OnDPeriodStruct</b>	<b>194</b>
<b>AIRTSP::SegmentCabinStruct</b>	<b>224</b>
<b>AIRTSP::SegmentStruct</b>	<b>241</b>
<b>TestFixture</b>	<b>299</b>
<b>AirlineScheduleTestSuite</b>	<b>140</b>

## 21 Class Index

### 21.1 Class List

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

<b>airtsp::Airline_T</b>	<b>139</b>
<b>AirlineScheduleTestSuite</b>	<b>140</b>
<b>AIRTSP::AIRTSP_Service</b> (Interface for the Airtsp Services )	<b>142</b>
<b>AIRTSP::AIRTSP_ServiceContext</b> (Class holding the context of the Airtsp services )	<b>146</b>
<b>BomAbstract</b>	<b>147</b>
<b>AIRTSP::BomDisplay</b> (Utility class to display AirTSP objects with a pretty format )	<b>147</b>
<b>CmdAbstract</b>	<b>148</b>
<b>airtsp::Date_T</b>	<b>149</b>
<b>AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition&lt; ScannerT &gt;</b>	<b>151</b>
<b>airtsp::SearchStringParser::definition&lt; ScannerT &gt;</b>	<b>158</b>
<b>AIRTSP::OnDParserHelper::OnDParser::definition&lt; ScannerT &gt;</b>	<b>162</b>
<b>AIRTSP::ScheduleParserHelper::doEndFlight</b>	<b>165</b>
<b>AIRTSP::OnDParserHelper::doEndOnD</b>	<b>166</b>
<b>AIRTSP::FacAIRTSPServiceContext</b> (Factory for the service context )	<b>168</b>
<b>AIRTSP::FacServiceAbstract</b>	<b>170</b>
<b>FacServiceAbstract</b>	<b>171</b>
<b>AIRTSP::FareFamilyStruct</b>	<b>171</b>
<b>FileNotFoundException</b>	<b>173</b>
<b>AIRTSP::FlagSaver</b>	<b>173</b>

<a href="#">AIRTSP::FlightPeriodFileParser</a>	174
<a href="#">AIRTSP::ScheduleParserHelper::FlightPeriodParser</a>	175
<a href="#">AIRTSP::FlightPeriodStruct</a>	176
<a href="#">grammar</a>	184
<a href="#">AIRTSP::InventoryGenerator</a>	184
<a href="#">KeyAbstract</a>	185
<a href="#">AIRTSP::LegCabinStruct</a>	185
<a href="#">AIRTSP::LegStruct</a>	187
<a href="#">AIRTSP::OnDInputFileNotFoundException</a>	190
<a href="#">AIRTSP::OnDParser (Class wrapping the parser entry point )</a>	190
<a href="#">AIRTSP::OnDParserHelper::OnDParser</a>	191
<a href="#">AIRTSP::OnDPeriodFileParser</a>	192
<a href="#">AIRTSP::OnDPeriodGenerator (Class handling the generation / instantiation of the O&amp;D-Period BOM )</a>	193
<a href="#">AIRTSP::OnDPeriodStruct</a>	194
<a href="#">AIRTSP::OriginDestinationSet (Class representing a simple sub-network )</a>	199
<a href="#">AIRTSP::OriginDestinationSetKey (Structure representing the key of a sub-network )</a>	204
<a href="#">ParserException</a>	206
<a href="#">AIRTSP::OnDParserHelper::ParserSemanticAction</a>	206
<a href="#">AIRTSP::ScheduleParserHelper::ParserSemanticAction</a>	208
<a href="#">airtsp::Passenger_T</a>	209
<a href="#">airtsp::Place_T</a>	211
<a href="#">AIRTSP::ReachableUniverse (Class representing the root of the schedule-related BOM tree )</a>	213
<a href="#">AIRTSP::ReachableUniverseKey (Structure representing the key of the schedule-related BOM tree root )</a>	217
<a href="#">AIRTSP::ScheduleInputFileNotFoundException</a>	220
<a href="#">AIRTSP::ScheduleParser</a>	220
<a href="#">airtsp::SearchString_T</a>	221
<a href="#">airtsp::SearchStringParser</a>	223
<a href="#">AIRTSP::SegmentCabinStruct</a>	224



<a href="#">AIRTSP::SegmentDateNotFoundException</a>	226
<a href="#">AIRTSP::SegmentPathGenerator</a> (Class handling the generation / instantiation of the network BOM )	227
<a href="#">AIRTSP::SegmentPathPeriod</a> (Class representing a segment/path )	228
<a href="#">AIRTSP::SegmentPathPeriodKey</a> (Structure representing the key of a segment/path )	234
<a href="#">AIRTSP::SegmentPathProvider</a> (Class building the travel solutions from airline schedules )	239
<a href="#">AIRTSP::SegmentPeriodHelper</a> (Class representing the actual business functions for an airline segment-period )	240
<a href="#">AIRTSP::SegmentStruct</a>	241
<a href="#">ServiceAbstract</a>	243
<a href="#">AIRTSP::ServiceAbstract</a>	243
<a href="#">AIRTSP::Simulator</a>	244
<a href="#">airtsp::store_adult_passenger_type</a>	245
<a href="#">airtsp::store_airline_code</a>	246
<a href="#">airtsp::store_airline_name</a>	247
<a href="#">airtsp::store_airline_sign</a>	248
<a href="#">airtsp::store_child_passenger_type</a>	249
<a href="#">airtsp::store_date</a>	250
<a href="#">airtsp::store_passenger_number</a>	251
<a href="#">airtsp::store_pet_passenger_type</a>	252
<a href="#">airtsp::store_place_element</a>	253
<a href="#">AIRTSP::OnDParserHelper::storeAirlineCode</a>	254
<a href="#">AIRTSP::ScheduleParserHelper::storeAirlineCode</a>	255
<a href="#">AIRTSP::ScheduleParserHelper::storeBoardingTime</a>	257
<a href="#">AIRTSP::ScheduleParserHelper::storeCapacity</a>	258
<a href="#">AIRTSP::OnDParserHelper::storeClassCode</a>	260
<a href="#">AIRTSP::ScheduleParserHelper::storeClasses</a>	261
<a href="#">AIRTSP::ScheduleParserHelper::storeDateRangeEnd</a>	263
<a href="#">AIRTSP::OnDParserHelper::storeDateRangeEnd</a>	264
<a href="#">AIRTSP::OnDParserHelper::storeDateRangeStart</a>	266

<a href="#">AIRTSP::ScheduleParserHelper::storeDateRangeStart</a>	267
<a href="#">AIRTSP::OnDParserHelper::storeDestination</a>	268
<a href="#">AIRTSP::ScheduleParserHelper::storeDow</a>	270
<a href="#">AIRTSP::ScheduleParserHelper::storeElapsedTime</a>	271
<a href="#">AIRTSP::OnDParserHelper::storeEndRangeTime</a>	273
<a href="#">AIRTSP::ScheduleParserHelper::storeFamilyCode</a>	274
<a href="#">AIRTSP::ScheduleParserHelper::storeFClasses</a>	275
<a href="#">AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey</a>	277
<a href="#">AIRTSP::ScheduleParserHelper::storeFlightNumber</a>	278
<a href="#">AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey</a>	280
<a href="#">AIRTSP::ScheduleParserHelper::storeLegBoardingPoint</a>	281
<a href="#">AIRTSP::ScheduleParserHelper::storeLegCabinCode</a>	283
<a href="#">AIRTSP::ScheduleParserHelper::storeLegOffPoint</a>	284
<a href="#">AIRTSP::ScheduleParserHelper::storeOffTime</a>	286
<a href="#">AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode</a>	287
<a href="#">AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber</a>	289
<a href="#">AIRTSP::OnDParserHelper::storeOrigin</a>	290
<a href="#">AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint</a>	292
<a href="#">AIRTSP::ScheduleParserHelper::storeSegmentCabinCode</a>	293
<a href="#">AIRTSP::ScheduleParserHelper::storeSegmentOffPoint</a>	295
<a href="#">AIRTSP::ScheduleParserHelper::storeSegmentSpecificity</a>	296
<a href="#">AIRTSP::OnDParserHelper::storeStartRangeTime</a>	298
<a href="#">StructAbstract</a>	299
<a href="#">TestFixture</a>	299
<a href="#">AIRTSP::TravelSolutionParser</a> (Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file )	300

## 22 File Index

### 22.1 File List

Here is a list of all files with brief descriptions:

<a href="#">airtsp/AIRTSP_Service.hpp</a>	302
<a href="#">airtsp/AIRTSP_Types.hpp</a>	305
<a href="#">airtsp/basic/BasConst.cpp</a>	307
<a href="#">airtsp/basic/BasConst_AIRTSP_Service.hpp</a>	309
<a href="#">airtsp/basic/BasConst_General.hpp</a>	311
<a href="#">airtsp/basic/BasParserTypes.hpp</a>	313
<a href="#">airtsp/batches/airtsp.cpp</a>	317
<a href="#">airtsp/batches/BookingRequestParser.cpp</a>	327
<a href="#">airtsp/batches/BookingRequestParser.hpp</a>	334
<a href="#">airtsp/bom/AirportList.hpp</a>	338
<a href="#">airtsp/bom/BomDisplay.cpp</a>	340
<a href="#">airtsp/bom/BomDisplay.hpp</a>	343
<a href="#">airtsp/bom/FareFamilyStruct.cpp</a>	345
<a href="#">airtsp/bom/FareFamilyStruct.hpp</a>	347
<a href="#">airtsp/bom/FlightPeriodStruct.cpp</a>	349
<a href="#">airtsp/bom/FlightPeriodStruct.hpp</a>	354
<a href="#">airtsp/bom/LegCabinStruct.cpp</a>	357
<a href="#">airtsp/bom/LegCabinStruct.hpp</a>	359
<a href="#">airtsp/bom/LegStruct.cpp</a>	361
<a href="#">airtsp/bom/LegStruct.hpp</a>	364
<a href="#">airtsp/bom/OnDPeriodStruct.cpp</a>	366
<a href="#">airtsp/bom/OnDPeriodStruct.hpp</a>	369
<a href="#">airtsp/bom/OriginDestinationSet.cpp</a>	371
<a href="#">airtsp/bom/OriginDestinationSet.hpp</a>	374
<a href="#">airtsp/bom/OriginDestinationSetKey.cpp</a>	377
<a href="#">airtsp/bom/OriginDestinationSetKey.hpp</a>	380
<a href="#">airtsp/bom/OriginDestinationSetTypes.hpp</a>	383
<a href="#">airtsp/bom/ReachableUniverse.cpp</a>	385
<a href="#">airtsp/bom/ReachableUniverse.hpp</a>	388

<a href="#">airtsp/bom/ReachableUniverseKey.cpp</a>	391
<a href="#">airtsp/bom/ReachableUniverseKey.hpp</a>	394
<a href="#">airtsp/bom/ReachableUniverseTypes.hpp</a>	397
<a href="#">airtsp/bom/SegmentCabinStruct.cpp</a>	399
<a href="#">airtsp/bom/SegmentCabinStruct.hpp</a>	401
<a href="#">airtsp/bom/SegmentPathPeriod.cpp</a>	403
<a href="#">airtsp/bom/SegmentPathPeriod.hpp</a>	409
<a href="#">airtsp/bom/SegmentPathPeriodKey.cpp</a>	413
<a href="#">airtsp/bom/SegmentPathPeriodKey.hpp</a>	416
<a href="#">airtsp/bom/SegmentPathPeriodTypes.hpp</a>	420
<a href="#">airtsp/bom/SegmentPeriodHelper.cpp</a>	422
<a href="#">airtsp/bom/SegmentPeriodHelper.hpp</a>	425
<a href="#">airtsp/bom/SegmentStruct.cpp</a>	427
<a href="#">airtsp/bom/SegmentStruct.hpp</a>	429
<a href="#">airtsp/command/InventoryGenerator.cpp</a>	431
<a href="#">airtsp/command/InventoryGenerator.hpp</a>	434
<a href="#">airtsp/command/OnDParser.cpp</a>	436
<a href="#">airtsp/command/OnDParser.hpp</a>	438
<a href="#">airtsp/command/OnDParserHelper.cpp</a>	440
<a href="#">airtsp/command/OnDParserHelper.hpp</a>	447
<a href="#">airtsp/command/OnDPeriodGenerator.cpp</a>	451
<a href="#">airtsp/command/OnDPeriodGenerator.hpp</a>	453
<a href="#">airtsp/command/ScheduleParser.cpp</a>	455
<a href="#">airtsp/command/ScheduleParser.hpp</a>	457
<a href="#">airtsp/command/ScheduleParserHelper.cpp</a>	459
<a href="#">airtsp/command/ScheduleParserHelper.hpp</a>	472
<a href="#">airtsp/command/SegmentPathGenerator.cpp</a>	477
<a href="#">airtsp/command/SegmentPathGenerator.hpp</a>	485
<a href="#">airtsp/command/SegmentPathProvider.cpp</a>	487

<a href="#">airtsp/command/SegmentPathProvider.hpp</a>	491
<a href="#">airtsp/command/Simulator.cpp</a>	493
<a href="#">airtsp/command/Simulator.hpp</a>	495
<a href="#">airtsp/command/TravelSolutionParser.cpp</a>	497
<a href="#">airtsp/command/TravelSolutionParser.hpp</a>	501
<a href="#">airtsp/config/airtsp-paths.hpp.in</a>	503
<a href="#">airtsp/factory/FacAIRTSPServiceContext.cpp</a>	505
<a href="#">airtsp/factory/FacAIRTSPServiceContext.hpp</a>	507
<a href="#">airtsp/factory/FacServiceAbstract.cpp</a>	509
<a href="#">airtsp/factory/FacServiceAbstract.hpp</a>	511
<a href="#">airtsp/service/AIRTSP_Service.cpp</a>	513
<a href="#">airtsp/service/AIRTSP_ServiceContext.cpp</a>	521
<a href="#">airtsp/service/AIRTSP_ServiceContext.hpp</a>	524
<a href="#">airtsp/service/ServiceAbstract.cpp</a>	527
<a href="#">airtsp/service/ServiceAbstract.hpp</a>	529
<a href="#">test/airtsp/AirlineScheduleTestSuite.cpp</a>	531
<a href="#">test/airtsp/AirlineScheduleTestSuite.hpp</a>	537

## 23 Directory Documentation

### 23.1 test/airtsp/ Directory Reference

#### Files

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

### 23.2 airtsp/ Directory Reference

#### Directories

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

**Files**

- file [AIRTSP\\_Service.hpp](#)
- file [AIRTSP\\_Types.hpp](#)

**23.3 airtsp/basic/ Directory Reference****Files**

- file [BasConst.cpp](#)
- file [BasConst\\_AIRTSP\\_Service.hpp](#)
- file [BasConst\\_General.hpp](#)
- file [BasParserTypes.hpp](#)

**23.4 airtsp/batches/ Directory Reference****Files**

- file [airtsp.cpp](#)
- file [BookingRequestParser.cpp](#)
- file [BookingRequestParser.hpp](#)

**23.5 airtsp/bom/ Directory Reference****Files**

- file [AirportList.hpp](#)
- file [BomDisplay.cpp](#)
- file [BomDisplay.hpp](#)
- file [FareFamilyStruct.cpp](#)
- file [FareFamilyStruct.hpp](#)
- file [FlightPeriodStruct.cpp](#)
- file [FlightPeriodStruct.hpp](#)
- file [LegCabinStruct.cpp](#)
- file [LegCabinStruct.hpp](#)
- file [LegStruct.cpp](#)
- file [LegStruct.hpp](#)
- file [OnDPeriodStruct.cpp](#)
- file [OnDPeriodStruct.hpp](#)
- file [OriginDestinationSet.cpp](#)
- file [OriginDestinationSet.hpp](#)
- file [OriginDestinationSetKey.cpp](#)
- file [OriginDestinationSetKey.hpp](#)
- file [OriginDestinationSetTypes.hpp](#)
- file [ReachableUniverse.cpp](#)
- file [ReachableUniverse.hpp](#)
- file [ReachableUniverseKey.cpp](#)
- file [ReachableUniverseKey.hpp](#)
- file [ReachableUniverseTypes.hpp](#)

- file [SegmentCabinStruct.cpp](#)
- file [SegmentCabinStruct.hpp](#)
- file [SegmentPathPeriod.cpp](#)
- file [SegmentPathPeriod.hpp](#)
- file [SegmentPathPeriodKey.cpp](#)
- file [SegmentPathPeriodKey.hpp](#)
- file [SegmentPathPeriodTypes.hpp](#)
- file [SegmentPeriodHelper.cpp](#)
- file [SegmentPeriodHelper.hpp](#)
- file [SegmentStruct.cpp](#)
- file [SegmentStruct.hpp](#)

## 23.6 airtsp/command/ Directory Reference

### Files

- file [InventoryGenerator.cpp](#)
- file [InventoryGenerator.hpp](#)
- file [OnDParser.cpp](#)
- file [OnDParser.hpp](#)
- file [OnDParserHelper.cpp](#)
- file [OnDParserHelper.hpp](#)
- file [OnDPeriodGenerator.cpp](#)
- file [OnDPeriodGenerator.hpp](#)
- file [ScheduleParser.cpp](#)
- file [ScheduleParser.hpp](#)
- file [ScheduleParserHelper.cpp](#)
- file [ScheduleParserHelper.hpp](#)
- file [SegmentPathGenerator.cpp](#)
- file [SegmentPathGenerator.hpp](#)
- file [SegmentPathProvider.cpp](#)
- file [SegmentPathProvider.hpp](#)
- file [Simulator.cpp](#)
- file [Simulator.hpp](#)
- file [TravelSolutionParser.cpp](#)
- file [TravelSolutionParser.hpp](#)

## 23.7 airtsp/config/ Directory Reference

### Files

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

## 23.8 airtsp/factory/ Directory Reference

### Files

- file [FacAIRTSPServiceContext.cpp](#)
- file [FacAIRTSPServiceContext.hpp](#)
- file [FacServiceAbstract.cpp](#)
- file [FacServiceAbstract.hpp](#)

## 23.9 airtsp/service/ Directory Reference

### Files

- file [AIRTSP\\_Service.cpp](#)
- file [AIRTSP\\_ServiceContext.cpp](#)
- file [AIRTSP\\_ServiceContext.hpp](#)
- file [ServiceAbstract.cpp](#)
- file [ServiceAbstract.hpp](#)

## 23.10 test/ Directory Reference

### Directories

- directory [airtsp](#)

## 24 Namespace Documentation

### 24.1 airtsp Namespace Reference

#### Classes

- struct [store\\_place\\_element](#)
- struct [store\\_date](#)
- struct [store\\_airline\\_sign](#)
- struct [store\\_airline\\_code](#)
- struct [store\\_airline\\_name](#)
- struct [store\\_passenger\\_number](#)
- struct [store\\_adult\\_passenger\\_type](#)
- struct [store\\_child\\_passenger\\_type](#)
- struct [store\\_pet\\_passenger\\_type](#)
- struct [SearchStringParser](#)
- struct [Place\\_T](#)
- struct [Date\\_T](#)
- struct [Airline\\_T](#)
- struct [Passenger\\_T](#)
- struct [SearchString\\_T](#)

#### Typedefs

- typedef std::vector< [Place\\_T](#) > [PlaceList\\_T](#)
- typedef std::vector< [Date\\_T](#) > [DateList\\_T](#)
- typedef std::vector< [Airline\\_T](#) > [AirlineList\\_T](#)
- typedef std::vector< [Passenger\\_T](#) > [PassengerList\\_T](#)

#### Functions

- [SearchString\\_T](#) [parseBookingRequest](#) (const std::string &iSearchString)



## Variables

- `boost::spirit::classic::int_parser< unsigned int, 10, 1, 1 >` [int1\\_p](#)
- `boost::spirit::classic::uint_parser< unsigned int, 10, 1, 1 >` [uint1\\_p](#)
- `boost::spirit::classic::uint_parser< unsigned int, 10, 1, 2 >` [uint1\\_2\\_p](#)
- `boost::spirit::classic::uint_parser< int, 10, 2, 2 >` [uint2\\_p](#)
- `boost::spirit::classic::uint_parser< int, 10, 2, 4 >` [uint2\\_4\\_p](#)
- `boost::spirit::classic::uint_parser< int, 10, 4, 4 >` [uint4\\_p](#)
- `boost::spirit::classic::uint_parser< int, 10, 1, 4 >` [uint1\\_4\\_p](#)

### 24.1.1 Typedef Documentation

#### 24.1.1.1 `typedef std::vector<Place_T> airtsp::PlaceList_T`

List of Place strucutres.

Definition at line 24 of file [BookingRequestParser.hpp](#).

#### 24.1.1.2 `typedef std::vector<Date_T> airtsp::DateList_T`

List of Date strucutres.

Definition at line 49 of file [BookingRequestParser.hpp](#).

#### 24.1.1.3 `typedef std::vector<Airline_T> airtsp::AirlineList_T`

List of Airline strucutres.

Definition at line 68 of file [BookingRequestParser.hpp](#).

#### 24.1.1.4 `typedef std::vector<Passenger_T> airtsp::PassengerList_T`

List of Passenger strucutres.

Definition at line 91 of file [BookingRequestParser.hpp](#).

### 24.1.2 Function Documentation

#### 24.1.2.1 `SearchString_T airtsp::parseBookingRequest (const std::string & iSearchString)`

Parse the booking request.

Sample guadeloupe rio de janeiro 07/22/2009 +aa -ua 2 adults 1 dog

Grammar: `search_string ::= places [dates] (preferred_airlines) (passengers) dates ::= board_date [off_date]`  
`places ::= [board_place] off_place board_place ::= place_elements off_place ::= place_elements place_`  
`elements ::= country | city | airport country ::= country_code | country_name city ::= city_code | city_name`  
`airport ::= airport_code | airport_name preferred_airlines ::= [+|-] airline_code | airline_name passen-`  
`gers ::= adult_number adult_description [child_number child_description] [pet_number pet_description]`  
`adult_description ::= 'adult' | 'adults' | 'pax' | 'passengers' child_description ::= 'child' | 'children' | 'kid'`  
`| 'kids' pet_description ::= 'dog' | 'dogs' | 'cat' | 'cats'`

### 24.1.3 Variable Documentation

#### 24.1.3.1 `boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> airtsp::int1_p`

1-digit-integer parser

Definition at line 203 of file [BookingRequestParser.cpp](#).

#### 24.1.3.2 `boost::spirit::classic::uint_parser<unsigned int, 10, 1, 1> airtsp::uint1_p`

1-digit-integer parser

Definition at line 205 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

#### 24.1.3.3 `boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> airtsp::uint1_2_p`

Up-to-2-digit-integer parser

Definition at line 207 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

#### 24.1.3.4 `boost::spirit::classic::uint_parser<int, 10, 2, 2> airtsp::uint2_p`

2-digit-integer parser

Definition at line 209 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

#### 24.1.3.5 `boost::spirit::classic::uint_parser<int, 10, 2, 4> airtsp::uint2_4_p`

Up-to-4-digit-integer parser

Definition at line 211 of file [BookingRequestParser.cpp](#).

#### 24.1.3.6 `boost::spirit::classic::uint_parser<int, 10, 4, 4> airtsp::uint4_p`

4-digit-integer parser

Definition at line 213 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

#### 24.1.3.7 `boost::spirit::classic::uint_parser<int, 10, 1, 4> airtsp::uint1_4_p`

Up-to-4-digit-integer parser

Definition at line 215 of file [BookingRequestParser.cpp](#).

## 24.2 AIRTSP Namespace Reference

### Namespaces

- namespace [ScheduleParserHelper](#)
- namespace [OnDParserHelper](#)

### Classes

- class [AIRTSP\\_Service](#)  
*Interface for the Airtsp Services.*
- class [SegmentDateNotFoundException](#)
- class [OnDInputFileNotFoundException](#)
- class [ScheduleInputFileNotFoundException](#)
- struct [FlagSaver](#)
- class [BomDisplay](#)  
*Utility class to display AirtSP objects with a pretty format.*
- struct [FareFamilyStruct](#)
- struct [FlightPeriodStruct](#)
- struct [LegCabinStruct](#)
- struct [LegStruct](#)
- struct [OnDPeriodStruct](#)
- class [OriginDestinationSet](#)  
*Class representing a simple sub-network.*
- struct [OriginDestinationSetKey](#)  
*Structure representing the key of a sub-network.*
- class [ReachableUniverse](#)  
*Class representing the root of the schedule-related BOM tree.*
- struct [ReachableUniverseKey](#)  
*Structure representing the key of the schedule-related BOM tree root.*
- struct [SegmentCabinStruct](#)
- class [SegmentPathPeriod](#)  
*Class representing a segment/path.*
- struct [SegmentPathPeriodKey](#)  
*Structure representing the key of a segment/path.*
- class [SegmentPeriodHelper](#)  
*Class representing the actual business functions for an airline segment-period.*
- struct [SegmentStruct](#)
- class [InventoryGenerator](#)
- class [OnDParser](#)  
*Class wrapping the parser entry point.*

- class [OnDPeriodFileParser](#)

- class [OnDPeriodGenerator](#)

*Class handling the generation / instantiation of the O&D-Period BOM.*

- class [ScheduleParser](#)

- class [FlightPeriodFileParser](#)

- class [SegmentPathGenerator](#)

*Class handling the generation / instantiation of the network BOM.*

- class [SegmentPathProvider](#)

*Class building the travel solutions from airline schedules.*

- class [Simulator](#)

- class [TravelSolutionParser](#)

*Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file.*

- class [FacAIRTSPServiceContext](#)

*Factory for the service context.*

- class [FacServiceAbstract](#)

- class [AIRTSP\\_ServiceContext](#)

*Class holding the context of the Airtsp services.*

- class [ServiceAbstract](#)

## Typedefs

- typedef boost::shared\_ptr< [AIRTSP\\_Service](#) > [AIRTSP\\_ServicePtr\\_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, 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< [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< [FareFamilyStruct](#) > [FareFamilyStructList\\_T](#)
- typedef std::vector< [LegCabinStruct](#) > [LegCabinStructList\\_T](#)
- typedef std::vector< [LegStruct](#) > [LegStructList\\_T](#)
- typedef std::list< [OriginDestinationSet](#) \* > [OriginDestinationSetList\\_T](#)

- typedef std::map< const stdair::MapKey\_T, OriginDestinationSet \* > OriginDestinationSetMap\_T
- typedef std::list< ReachableUniverse \* > ReachableUniverseList\_T
- typedef std::map< const stdair::MapKey\_T, ReachableUniverse \* > ReachableUniverseMap\_T
- typedef std::vector< SegmentCabinStruct > SegmentCabinStructList\_T
- typedef std::list< SegmentPathPeriod \* > SegmentPathPeriodList\_T
- typedef std::multimap< const stdair::MapKey\_T, SegmentPathPeriod \* > SegmentPathPeriodMultimap\_T
- typedef std::vector< const SegmentPathPeriod \* > SegmentPathPeriodLightList\_T
- typedef std::vector< SegmentPathPeriodLightList\_T > SegmentPathPeriodListList\_T
- typedef std::vector< stdair::DateOffset\_T > DateOffsetList\_T
- typedef std::vector< SegmentStruct > SegmentStructList\_T

## Functions

- const stdair::Duration\_T MINIMUM\_TIME\_BETWEEN\_REQUEST\_AND\_DEPARTURE (4, 0, 0)
- template void OriginDestinationSet::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void OriginDestinationSet::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)
- template void OriginDestinationSetKey::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void OriginDestinationSetKey::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)
- template void ReachableUniverse::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void ReachableUniverse::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)
- template void ReachableUniverseKey::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void ReachableUniverseKey::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)
- template void SegmentPathPeriod::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void SegmentPathPeriod::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)
- template void SegmentPathPeriodKey::serialize< ba::text\_oarchive > (ba::text\_oarchive &, unsigned int)
- template void SegmentPathPeriodKey::serialize< ba::text\_iarchive > (ba::text\_iarchive &, unsigned int)

## Variables

- const int DEFAULT\_NUMBER\_OF\_DRAWS\_FOR\_MC\_SIMULATION = 100000
- const stdair::Duration\_T MINIMUM\_TIME\_BETWEEN\_REQUEST\_AND\_DEPARTURE

### 24.2.1 Typedef Documentation

#### 24.2.1.1 typedef boost::shared\_ptr<AIRTSP\_Service> AIRTSP::AIRTSP\_ServicePtr\_T

(Smart) Pointer on the Airtsp service handler.

Definition at line 62 of file [AIRTSP\\_Types.hpp](#).

**24.2.1.2 typedef char AIRTSP::char\_t**

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

**24.2.1.3 typedef boost::spirit::classic::file\_iterator<char\_t> AIRTSP::iterator\_t**

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

**24.2.1.4 typedef boost::spirit::classic::scanner<iterator\_t> AIRTSP::scanner\_t**

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

**24.2.1.5 typedef boost::spirit::classic::rule<scanner\_t> AIRTSP::rule\_t**

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

**24.2.1.6 typedef boost::spirit::classic::int\_parser<unsigned int, 10, 1, 1> AIRTSP::int1\_p\_t**

1-digit-integer parser

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

**24.2.1.7 typedef boost::spirit::classic::uint\_parser<unsigned int, 10, 2, 2> AIRTSP::uint2\_p\_t**

2-digit-integer parser

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

**24.2.1.8 typedef boost::spirit::classic::uint\_parser<unsigned int, 10, 4, 4> AIRTSP::uint4\_p\_t**

4-digit-integer parser

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

**24.2.1.9 typedef boost::spirit::classic::uint\_parser<unsigned int, 10, 1, 4> AIRTSP::uint1\_4\_p\_t**

Up-to-4-digit-integer parser

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

**24.2.1.10 typedef boost::spirit::classic::chset<char\_t> AIRTSP::chset\_t**

character set

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

**24.2.1.11** `typedef boost::spirit::classic::impl::loop_traits<chset_t, unsigned int, unsigned int>::type AIRTSP::repeat_p_t`

(Repeating) sequence of a given number of characters: repeat\_p(min, max)

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

**24.2.1.12** `typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int>  
AIRTSP::bounded2_p_t`

Bounded-number-of-integers parser

Definition at line 66 of file [BasParserTypes.hpp](#).

**24.2.1.13** `typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int>  
AIRTSP::bounded4_p_t`

Definition at line 67 of file [BasParserTypes.hpp](#).

**24.2.1.14** `typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>  
AIRTSP::bounded1_4_p_t`

Definition at line 68 of file [BasParserTypes.hpp](#).

**24.2.1.15** `typedef std::set<stdair::AirportCode_T> AIRTSP::AirportList_T`

Define lists of Airport Codes.

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

**24.2.1.16** `typedef std::vector<stdair::AirportCode_T> AIRTSP::AirportOrderedList_T`

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

**24.2.1.17** `typedef std::vector<FareFamilyStruct> AIRTSP::FareFamilyStructList_T`

List of FareFamily-Detail structures.

Definition at line 34 of file [FareFamilyStruct.hpp](#).

**24.2.1.18** `typedef std::vector<LegCabinStruct> AIRTSP::LegCabinStructList_T`

List of LegCabin-Detail structures.

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

**24.2.1.19** `typedef std::vector<LegStruct> AIRTSP::LegStructList_T`

List of Leg strucutres.

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

**24.2.1.20** `typedef std::list<OriginDestinationSet*> AIRTSP::OriginDestinationSetList_T`

Define the [OriginDestinationSet](#) list.

Definition at line 18 of file [OriginDestinationSetTypes.hpp](#).

**24.2.1.21** `typedef std::map<const stdair::MapKey_T, OriginDestinationSet*>  
AIRTSP::OriginDestinationSetMap_T`

Define the [OriginDestinationSet](#) map.

Definition at line 25 of file [OriginDestinationSetTypes.hpp](#).

**24.2.1.22** `typedef std::list<ReachableUniverse*> AIRTSP::ReachableUniverseList_T`

Define the reachable-universe list.

Definition at line 18 of file [ReachableUniverseTypes.hpp](#).

**24.2.1.23** `typedef std::map<const stdair::MapKey_T, ReachableUniverse*>  
AIRTSP::ReachableUniverseMap_T`

Define the reachable-universe map.

Definition at line 25 of file [ReachableUniverseTypes.hpp](#).

**24.2.1.24** `typedef std::vector<SegmentCabinStruct> AIRTSP::SegmentCabinStructList_T`

List of SegmentCabin-Detail strucutres.

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

**24.2.1.25** `typedef std::list<SegmentPathPeriod*> AIRTSP::SegmentPathPeriodList_T`

Define the segment path list.

Definition at line 20 of file [SegmentPathPeriodTypes.hpp](#).

**24.2.1.26** `typedef std::multimap<const stdair::MapKey_T, SegmentPathPeriod*>  
AIRTSP::SegmentPathPeriodMultimap_T`

Define the segment path map.

Definition at line 27 of file [SegmentPathPeriodTypes.hpp](#).



**24.2.1.27** `typedef std::vector<const SegmentPathPeriod*>  
AIRTSP::SegmentPathPeriodLightList_T`

Define the (temporary) containers of segment path period.

Definition at line 30 of file [SegmentPathPeriodTypes.hpp](#).

**24.2.1.28** `typedef std::vector<SegmentPathPeriodLightList_T>  
AIRTSP::SegmentPathPeriodListList_T`

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

**24.2.1.29** `typedef std::vector<stdair::DateOffset_T> AIRTSP::DateOffsetList_T`

Define the vector of boarding date offsets of the member segments of a segment path compare to the boarding date of the first segment.

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

**24.2.1.30** `typedef std::vector<SegmentStruct> AIRTSP::SegmentStructList_T`

List of Segment strucutres.

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

## 24.2.2 Function Documentation

**24.2.2.1** `const stdair::Duration_T AIRTSP::MINIMUM_TIME_BETWEEN_REQUEST_AND_  
DEPARTURE (4, 0, 0)`

Default value for the minimum time between the request and the departure time.

**24.2.2.2** `template void AIRTSP::OriginDestinationSet::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`

**24.2.2.3** `template void AIRTSP::OriginDestinationSet::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`

**24.2.2.4** `template void AIRTSP::OriginDestinationSetKey::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`

- 24.2.2.5** `template void AIRTSP::OriginDestinationSetKey::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`
- 24.2.2.6** `template void AIRTSP::ReachableUniverse::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`
- 24.2.2.7** `template void AIRTSP::ReachableUniverse::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`
- 24.2.2.8** `template void AIRTSP::ReachableUniverseKey::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`
- 24.2.2.9** `template void AIRTSP::ReachableUniverseKey::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`
- 24.2.2.10** `template void AIRTSP::SegmentPathPeriod::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`
- 24.2.2.11** `template void AIRTSP::SegmentPathPeriod::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`
- 24.2.2.12** `template void AIRTSP::SegmentPathPeriodKey::serialize< ba::text_oarchive >  
(ba::text_oarchive &, unsigned int)`
- 24.2.2.13** `template void AIRTSP::SegmentPathPeriodKey::serialize< ba::text_iarchive >  
(ba::text_iarchive &, unsigned int)`

### 24.2.3 Variable Documentation

#### 24.2.3.1 `const int AIRTSP::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000`

Default value for the number of draws within the Monte-Carlo Integration algorithm.

Definition at line 8 of file [BasConst.cpp](#).

#### 24.2.3.2 `const std::duration<T> AIRTSP::MINIMUM_TIME_BETWEEN_REQUEST_AND_DEPARTURE`

Default value for the minimum time between the request and the departure time.

## 24.3 AIRTSP::OnDParserHelper Namespace Reference

### Classes

- struct [ParserSemanticAction](#)
- struct [storeOrigin](#)
- struct [storeDestination](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeStartRangeTime](#)
- struct [storeEndRangeTime](#)
- struct [storeAirlineCode](#)
- struct [storeClassCode](#)
- struct [doEndOnD](#)
- struct [OnDParser](#)

### Functions

- [chset\\_t alpha\\_cap\\_set\\_p](#) ("A-Z")
- [repeat\\_p\\_t airport\\_p](#) ([chset\\_t](#)("0-9A-Z").derived(), 3, 3)
- [repeat\\_p\\_t airline\\_code\\_p](#) ([alpha\\_cap\\_set\\_p](#).derived(), 2, 3)
- [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)
- [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 class\\_code\\_p](#) ("A-Z")

### Variables

- [uint2\\_p\\_t uint2\\_p](#)
- [uint4\\_p\\_t uint4\\_p](#)
- [uint1\\_4\\_p\\_t uint1\\_4\\_p](#)

### 24.3.1 Function Documentation

#### 24.3.1.1 chset\_t AIRTSP::OnDParserHelper::alpha\_cap\_set\_p ("A-Z")

Sequence of (capital) alphabetic characters: chset\_p("A-Z")

#### 24.3.1.2 repeat\_p\_t AIRTSP::OnDParserHelper::airport\_p (chset\_t("0-9A-Z").derived(), 3, 3)

Airport Parser: repeat\_p(3)[chset\_p("0-9A-Z")]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.3 repeat\_p\_t AIRTSP::OnDParserHelper::airline\_code\_p (alpha\_cap\_set\_p.derived(), 2, 3)

Airline Code Parser: repeat\_p(2,3)[chset\_p("0-9A-Z")]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.4 bounded4\_p\_t AIRTSP::OnDParserHelper::year\_p (uint4\_p.derived(), 2000u, 2099u)

Year Parser: limit\_d(2000u, 2099u)[uint4\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.5 bounded2\_p\_t AIRTSP::OnDParserHelper::month\_p (uint2\_p.derived(), 1u, 12u)

Month Parser: limit\_d(1u, 12u)[uint2\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.6 bounded2\_p\_t AIRTSP::OnDParserHelper::day\_p (uint2\_p.derived(), 1u, 31u)

Day Parser: limit\_d(1u, 31u)[uint2\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.7 bounded2\_p\_t AIRTSP::OnDParserHelper::hours\_p (uint2\_p.derived(), 0u, 23u)

Hour Parser: limit\_d(0u, 23u)[uint2\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 24.3.1.8 bounded2\_p\_t AIRTSP::OnDParserHelper::minutes\_p (uint2\_p.derived(), 0u, 59u)

Minute Parser: limit\_d(0u, 59u)[uint2\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

**24.3.1.9 bounded2\_p\_t AIRTSP::OnDParserHelper::seconds\_p (uint2\_p. *derived*(), 0u, 59u)**

Second Parser: limit\_d(0u, 59u)[uint2\_p]

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).**24.3.1.10 chset\_t AIRTSP::OnDParserHelper::class\_code\_p ("A-Z")**

Class Code Parser: chset\_p("A-Z")

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).**24.3.2 Variable Documentation****24.3.2.1 uint2\_p\_t AIRTSP::OnDParserHelper::uint2\_p**

2-digit-integer parser

Definition at line 215 of file [OnDParserHelper.cpp](#).**24.3.2.2 uint4\_p\_t AIRTSP::OnDParserHelper::uint4\_p**

4-digit-integer parser

Definition at line 218 of file [OnDParserHelper.cpp](#).**24.3.2.3 uint1\_4\_p\_t AIRTSP::OnDParserHelper::uint1\_4\_p**

Up-to-4-digit-integer parser

Definition at line 221 of file [OnDParserHelper.cpp](#).**24.4 AIRTSP::ScheduleParserHelper Namespace Reference****Classes**

- struct [ParserSemanticAction](#)
- struct [storeAirlineCode](#)
- struct [storeFlightNumber](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeDow](#)
- struct [storeLegBoardingPoint](#)
- struct [storeLegOffPoint](#)
- struct [storeOperatingAirlineCode](#)
- struct [storeOperatingFlightNumber](#)
- struct [storeBoardingTime](#)
- struct [storeOffTime](#)
- struct [storeElapsedTime](#)
- struct [storeLegCabinCode](#)
- struct [storeCapacity](#)

- struct [storeSegmentSpecificity](#)
- struct [storeSegmentBoardingPoint](#)
- struct [storeSegmentOffPoint](#)
- struct [storeSegmentCabinCode](#)
- struct [storeClasses](#)
- struct [storeFamilyCode](#)
- struct [storeFRAT5CurveKey](#)
- struct [storeFFDisutilityCurveKey](#)
- 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 key\\_p](#) ([chset\\_t](#)("0-9A-Z").[derived](#)(), 1, 10)
- [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](#)

### 24.4.1 Function Documentation

#### 24.4.1.1 [repeat\\_p\\_t AIRTSP::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 [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

#### 24.4.1.2 bounded1\_4\_p\_t AIRTSP::ScheduleParserHelper::flight\_number\_p (uint1\_4\_p. *derived()*, 0u, 9999u)

Flight Number Parser: limit\_d(0u, 9999u)[uint1\_4\_p]

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

#### 24.4.1.3 bounded4\_p\_t AIRTSP::ScheduleParserHelper::year\_p (uint4\_p.*derived()*, 2000u, 2099u)

Year Parser: limit\_d(2000u, 2099u)[uint4\_p]

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

#### 24.4.1.4 bounded2\_p\_t AIRTSP::ScheduleParserHelper::month\_p (uint2\_p.*derived()*, 1u, 12u)

Month Parser: limit\_d(1u, 12u)[uint2\_p]

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

#### 24.4.1.5 bounded2\_p\_t AIRTSP::ScheduleParserHelper::day\_p (uint2\_p.*derived()*, 1u, 31u)

Day Parser: limit\_d(1u, 31u)[uint2\_p]

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

#### 24.4.1.6 repeat\_p\_t AIRTSP::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 [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

#### 24.4.1.7 repeat\_p\_t AIRTSP::ScheduleParserHelper::airport\_p (chset\_t("0-9A-Z").*derived()*, 3, 3)

Airport Parser: repeat\_p(3)[chset\_p("0-9A-Z")]

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

#### 24.4.1.8 bounded2\_p\_t AIRTSP::ScheduleParserHelper::hours\_p (uint2\_p.*derived()*, 0u, 23u)

Hour Parser: limit\_d(0u, 23u)[uint2\_p]

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

**24.4.1.9 bounded2\_p\_t AIRTSP::ScheduleParserHelper::minutes\_p (uint2\_p. *derived*(), 0u, 59u)**

Minute Parser: limit\_d(0u, 59u)[uint2\_p]

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.1.10 bounded2\_p\_t AIRTSP::ScheduleParserHelper::seconds\_p (uint2\_p. *derived*(), 0u, 59u)**

Second Parser: limit\_d(0u, 59u)[uint2\_p]

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.1.11 chset\_t AIRTSP::ScheduleParserHelper::cabin\_code\_p ("A-Z")**

Cabin Code Parser: chset\_p("A-Z")

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.1.12 repeat\_p\_t AIRTSP::ScheduleParserHelper::key\_p (chset\_t("0-9A-Z").*derived*(), 1, 10)**

Key Parser: repeat\_p(1,10)[chset\_p("0-9A-Z")]

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.1.13 repeat\_p\_t AIRTSP::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 [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.2 Variable Documentation****24.4.2.1 int1\_p\_t AIRTSP::ScheduleParserHelper::int1\_p**

1-digit-integer parser

Definition at line 473 of file [ScheduleParserHelper.cpp](#).Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).**24.4.2.2 uint2\_p\_t AIRTSP::ScheduleParserHelper::uint2\_p**

2-digit-integer parser

Definition at line 476 of file [ScheduleParserHelper.cpp](#).



#### 24.4.2.3 uint4\_p\_t AIRTSP::ScheduleParserHelper::uint4\_p

4-digit-integer parser

Definition at line 479 of file [ScheduleParserHelper.cpp](#).

#### 24.4.2.4 uint1\_4\_p\_t AIRTSP::ScheduleParserHelper::uint1\_4\_p

Up-to-4-digit-integer parser

Definition at line 482 of file [ScheduleParserHelper.cpp](#).

#### 24.4.2.5 int1\_p\_t AIRTSP::ScheduleParserHelper::family\_code\_p

Family code parser

Definition at line 518 of file [ScheduleParserHelper.cpp](#).

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

## 24.5 boost Namespace Reference

Forward declarations.

### Namespaces

- namespace [serialization](#)

### 24.5.1 Detailed Description

Forward declarations.

## 24.6 boost::serialization Namespace Reference

## 24.7 stdair Namespace Reference

Forward declarations.

### 24.7.1 Detailed Description

Forward declarations.

# 25 Class Documentation

## 25.1 airtsp::Airline\_T Struct Reference

```
#include <airtsp/batches/BookingRequestParser.hpp>
```

## Public Member Functions

- [Airline\\_T](#) ()
- void [display](#) () const

## Public Attributes

- bool [\\_isPreferred](#)
- std::string [\\_name](#)
- std::string [\\_code](#)

### 25.1.1 Detailed Description

Airline.

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

### 25.1.2 Constructor & Destructor Documentation

#### 25.1.2.1 airtsp::Airline\_T::Airline\_T () [inline]

Constructor.

Definition at line 58 of file [BookingRequestParser.hpp](#).

### 25.1.3 Member Function Documentation

#### 25.1.3.1 void airtsp::Airline\_T::display () const [inline]

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

References [\\_code](#), [\\_isPreferred](#), and [\\_name](#).

Referenced by [airtsp::SearchString\\_T::display\(\)](#).

### 25.1.4 Member Data Documentation

#### 25.1.4.1 bool airtsp::Airline\_T::\_isPreferred

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

Referenced by [display\(\)](#), and [airtsp::store\\_airline\\_sign::operator\(\)](#).

#### 25.1.4.2 std::string airtsp::Airline\_T::\_name

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

Referenced by [display\(\)](#), and [airtsp::store\\_airline\\_name::operator\(\)](#).

### 25.1.4.3 std::string airtsp::Airline\_T::\_code

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

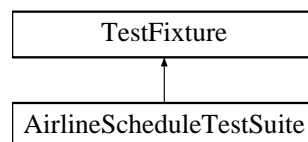
Referenced by [display\(\)](#), and [airtsp::store\\_airline\\_code::operator\(\)\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.hpp](#)

## 25.2 AirlineScheduleTestSuite Class Reference

`#include <test/airtsp/AirlineScheduleTestSuite.hpp>`Inheritance diagram for AirlineScheduleTestSuite::



### Public Member Functions

- void [externalMemoryManagement](#) ()
- void [scheduleParsing](#) ()
- [AirlineScheduleTestSuite](#) ()

### Protected Attributes

- std::stringstream [\\_describeKey](#)

### 25.2.1 Detailed Description

Definition at line 6 of file [AirlineScheduleTestSuite.hpp](#).

### 25.2.2 Constructor & Destructor Documentation

#### 25.2.2.1 AirlineScheduleTestSuite::AirlineScheduleTestSuite ()

Constructor.

### 25.2.3 Member Function Documentation

#### 25.2.3.1 void AirlineScheduleTestSuite::externalMemoryManagement ()

Test the Optimisation functionality.

The code is aimed at testing the initialization of airline inventory-related objects which are mainly implemented in the [stdair](#) library. That means the memory allocation of these objects will be managed by the calling project and not by the called project.

### 25.2.3.2 void AirlineScheduleTestSuite::scheduleParsing ()

## 25.2.4 Member Data Documentation

### 25.2.4.1 std::stringstream AirlineScheduleTestSuite::\_describeKey [protected]

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

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

- test/airtsp/[AirlineScheduleTestSuite.hpp](#)

## 25.3 AIRTSP::AIRTSP\_Service Class Reference

Interface for the Airtsp Services.

```
#include <airtsp/AIRTSP_Service.hpp>
```

### Public Member Functions

- [AIRTSP\\_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [AIRTSP\\_Service](#) (const stdair::BasLogParams &)
- [AIRTSP\\_Service](#) (stdair::STDAIR\_ServicePtr\_T ioSTDAIR\_ServicePtr)
- void [parseAndLoad](#) (const stdair::ScheduleFilePath &)
- void [parseAndLoad](#) (const stdair::ScheduleFilePath &, const stdair::ODFilePath &)
- [~AIRTSP\\_Service](#) ()
- void [buildSampleBom](#) ()
- void [clonePersistentBom](#) ()
- void [buildComplementaryLinks](#) (stdair::BomRoot &)
- void [buildSegmentPathList](#) (stdair::TravelSolutionList\_T &, const stdair::BookingRequestStruct &)
- void [simulate](#) ()
- std::string [jsonExportFlightDateObjects](#) (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

### 25.3.1 Detailed Description

Interface for the Airtsp Services.

Definition at line 32 of file [AIRTSP\\_Service.hpp](#).

### 25.3.2 Constructor & Destructor Documentation

#### 25.3.2.1 AIRTSP::AIRTSP\_Service::AIRTSP\_Service (const stdair::BasLogParams & iLogParams, const stdair::BasDBParams & iDBParams)

Constructor.

The initAirtspService() 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:

**const** stdair::BasLogParams& Parameters for the output log stream.

**const** stdair::BasDBParams& Parameters for the database access.

Definition at line 62 of file [AIRTSP\\_Service.cpp](#).

#### 25.3.2.2 AIRTSP::AIRTSP\_Service::AIRTSP\_Service (const stdair::BasLogParams & iLogParams)

Constructor.

The initAirtspService() 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:

**const** stdair::BasLogParams& Parameters for the output log stream.

Definition at line 42 of file [AIRTSP\\_Service.cpp](#).

#### 25.3.2.3 AIRTSP::AIRTSP\_Service::AIRTSP\_Service (stdair::STDAIR\_ServicePtr\_T ioSTDAIR\_ServicePtr)

Constructor.

The initAirtspService() 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 [AIRTSP\\_Service](#) is itself being initialised by another library service such as SIMCRS\_Service).

##### Parameters:

**stdair::STDAIR\_ServicePtr\_T** Reference on the STDAIR service.

Definition at line 84 of file [AIRTSP\\_Service.cpp](#).

#### 25.3.2.4 AIRTSP::AIRTSP\_Service::~~AIRTSP\_Service ()

Destructor.

Definition at line 100 of file [AIRTSP\\_Service.cpp](#).

### 25.3.3 Member Function Documentation

#### 25.3.3.1 void AIRTSP::AIRTSP\_Service::parseAndLoad (const stdair::ScheduleFilePath & iScheduleInputFilePath)

Parse the schedule input file and load it into memory.

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

##### Parameters:

**const** stdair::ScheduleFilePath& Filename of the input schedule file.

Definition at line 178 of file [AIRTSP\\_Service.cpp](#).

References [buildComplementaryLinks\(\)](#), [clonePersistentBom\(\)](#), and [AIRTSP::ScheduleParser::generateInventories\(\)](#).

Referenced by [main\(\)](#), and [parseAndLoad\(\)](#).

#### 25.3.3.2 void AIRTSP::AIRTSP\_Service::parseAndLoad (const stdair::ScheduleFilePath & iScheduleInputFilePath, const stdair::ODFilePath & iODInputFilePath)

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:

**const** stdair::ScheduleFilePath& Filename of the input schedule file.

**const** stdair::ODFilePath& Filename of the input O&D file.

Definition at line 230 of file [AIRTSP\\_Service.cpp](#).

References [buildComplementaryLinks\(\)](#), [AIRTSP::OnDParser::generateOnDPeriods\(\)](#), and [parseAndLoad\(\)](#).

#### 25.3.3.3 void AIRTSP::AIRTSP\_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 287 of file [AIRTSP\\_Service.cpp](#).

References [buildComplementaryLinks\(\)](#), and [clonePersistentBom\(\)](#).

Referenced by [main\(\)](#).

**25.3.3.4 void AIRTSP::AIRTSP\_Service::clonePersistentBom ()**

Clone the persistent BOM object.

Definition at line 344 of file [AIRTSP\\_Service.cpp](#).

References [buildComplementaryLinks\(\)](#).

Referenced by [buildSampleBom\(\)](#), and [parseAndLoad\(\)](#).

**25.3.3.5 void AIRTSP::AIRTSP\_Service::buildComplementaryLinks (stdair::BomRoot & ioBomRoot)**

Build all the complementary links in the given bom root object.

Definition at line 384 of file [AIRTSP\\_Service.cpp](#).

References [AIRTSP::SegmentPathGenerator::createSegmentPathNetwork\(\)](#).

Referenced by [buildSampleBom\(\)](#), [clonePersistentBom\(\)](#), and [parseAndLoad\(\)](#).

**25.3.3.6 void AIRTSP::AIRTSP\_Service::buildSegmentPathList (stdair::TravelSolutionList\_T & ioTravelSolutionList, const stdair::BookingRequestStruct & iBookingRequest)**

Calculate and return a list of travel solutions corresponding to a given product demand.

Definition at line 498 of file [AIRTSP\\_Service.cpp](#).

Referenced by [main\(\)](#).

**25.3.3.7 void AIRTSP::AIRTSP\_Service::simulate ()**

Perform a small simulation, which uses the Customer Choice Model (CCM).

Currently, that method does nothing.

Definition at line 470 of file [AIRTSP\\_Service.cpp](#).

**25.3.3.8 std::string AIRTSP::AIRTSP\_Service::jsonExportFlightDateObjects (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-period corresponding to the parameters given as input.

**Parameters:**

**const** stdair::AirlineCode\_T& Airline code of the flight to dump.

**const** stdair::FlightNumber\_T& Flight number of the flight to dump.

**const** stdair::Date\_T& Departure date of a flight within the flight period to dump.

**Returns:**

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

Definition at line 401 of file [AIRTSP\\_Service.cpp](#).

**25.3.3.9 std::string AIRTSP::AIRTSP\_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 425 of file [AIRTSP\\_Service.cpp](#).

**25.3.3.10 std::string AIRTSP::AIRTSP\_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-period corresponding to the parameters given as input.

**Parameters:**

**const** stdair::AirlineCode\_T& Airline code of the flight period to display.

**const** stdair::FlightNumber\_T& Flight number of the flight to display.

**const** stdair::Date\_T& Departure date of a flight within the flight-period to display.

**Returns:**

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

Definition at line 447 of file [AIRTSP\\_Service.cpp](#).

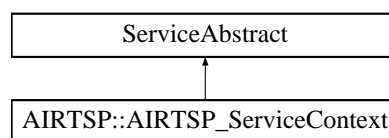
The documentation for this class was generated from the following files:

- [airtsp/AIRTSP\\_Service.hpp](#)
- [airtsp/service/AIRTSP\\_Service.cpp](#)

**25.4 AIRTSP::AIRTSP\_ServiceContext Class Reference**

Class holding the context of the Airtsp services.

#include <airtsp/service/AIRTSP\_ServiceContext.hpp> Inheritance diagram for AIRTSP::AIRTSP\_ServiceContext::

**Friends**

- class [AIRTSP\\_Service](#)
- class [FacAIRTSPServiceContext](#)



### 25.4.1 Detailed Description

Class holding the context of the Airtsp services.

Definition at line 22 of file [AIRTSP\\_ServiceContext.hpp](#).

### 25.4.2 Friends And Related Function Documentation

#### 25.4.2.1 friend class AIRTSP\_Service [friend]

The [AIRTSP\\_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 28 of file [AIRTSP\\_ServiceContext.hpp](#).

#### 25.4.2.2 friend class FacAIRTSPServiceContext [friend]

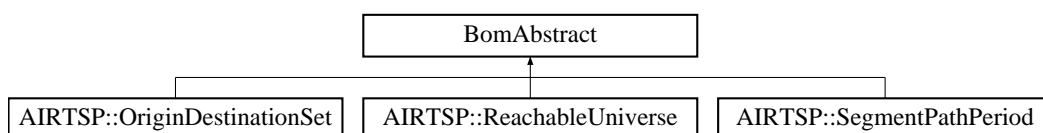
Definition at line 29 of file [AIRTSP\\_ServiceContext.hpp](#).

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

- [airtsp/service/AIRTSP\\_ServiceContext.hpp](#)
- [airtsp/service/AIRTSP\\_ServiceContext.cpp](#)

## 25.5 BomAbstract Class Reference

Inheritance diagram for BomAbstract::



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

- [airtsp/bom/SegmentPathPeriod.hpp](#)
- [airtsp/bom/OriginDestinationSet.hpp](#)
- [airtsp/bom/ReachableUniverse.hpp](#)

## 25.6 AIRTSP::BomDisplay Class Reference

Utility class to display AirTSP objects with a pretty format.

```
#include <airtsp/bom/BomDisplay.hpp>
```

### Static Public Member Functions

- static std::string [csvDisplay](#) (const stdair::BomRoot &)
- static void [csvDisplay](#) (std::ostream &, const [ReachableUniverse](#) &)

### 25.6.1 Detailed Description

Utility class to display AirTSP objects with a pretty format.

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

### 25.6.2 Member Function Documentation

#### 25.6.2.1 `std::string AIRTSP::BomDisplay::csvDisplay (const stdair::BomRoot & iBomRoot) [static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

##### Parameters:

*std::ostream&* Output stream in which the BOM tree should be logged/dumped.

*const* *stdair::EventQueue&* Root of the BOM tree to be displayed.

Definition at line 43 of file [BomDisplay.cpp](#).

#### 25.6.2.2 `void AIRTSP::BomDisplay::csvDisplay (std::ostream & oStream, const ReachableUniverse & iReachableUniverse) [static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

##### Parameters:

*std::ostream&* Output stream in which the BOM tree should be logged/dumped.

*const* [ReachableUniverse](#)& Root of the BOM tree to be displayed.

Definition at line 81 of file [BomDisplay.cpp](#).

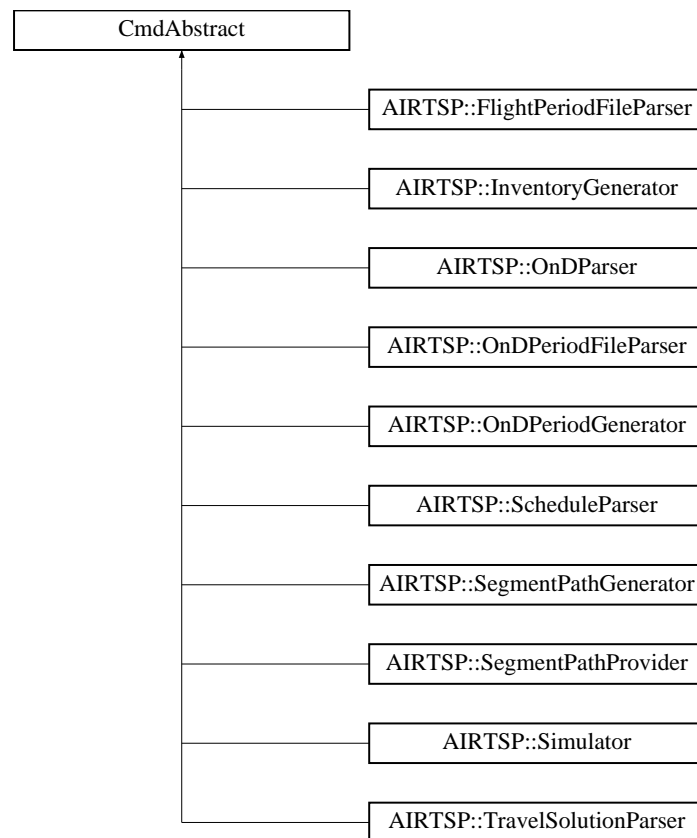
References [AIRTSP::ReachableUniverse::toString\(\)](#).

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

- [airtsp/bom/BomDisplay.hpp](#)
- [airtsp/bom/BomDisplay.cpp](#)

## 25.7 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract::



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

- [airtsp/command/Simulator.hpp](#)
- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/SegmentPathProvider.hpp](#)
- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/SegmentPathGenerator.hpp](#)
- [airtsp/command/InventoryGenerator.hpp](#)
- [airtsp/command/OnDPeriodGenerator.hpp](#)
- [airtsp/command/TravelSolutionParser.hpp](#)
- [airtsp/command/ScheduleParser.hpp](#)
- [airtsp/command/OnDParser.hpp](#)

## 25.8 airtsp::Date\_T Struct Reference

```
#include <airtsp/batches/BookingRequestParser.hpp>
```

### Public Member Functions

- [Date\\_T \(\)](#)
- void [display \(\)](#) const
- boost::gregorian::date [getDate \(\)](#) const

## Public Attributes

- [boost::gregorian::date \\_date](#)
- [unsigned int \\_reldays](#)
- [unsigned int \\_day](#)
- [unsigned int \\_month](#)
- [unsigned int \\_year](#)

### 25.8.1 Detailed Description

Date.

Definition at line 27 of file [BookingRequestParser.hpp](#).

### 25.8.2 Constructor & Destructor Documentation

#### 25.8.2.1 airtsp::Date\_T::Date\_T () [\[inline\]](#)

Constructor.

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

### 25.8.3 Member Function Documentation

#### 25.8.3.1 void airtsp::Date\_T::display () const [\[inline\]](#)

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

References [\\_date](#), [\\_day](#), [\\_month](#), [\\_reldays](#), and [\\_year](#).

Referenced by [airtsp::SearchString\\_T::display\(\)](#).

#### 25.8.3.2 boost::gregorian::date airtsp::Date\_T::getDate () const [\[inline\]](#)

Set the date from the staging details.

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

References [\\_day](#), [\\_month](#), and [\\_year](#).

Referenced by [airtsp::store\\_date::operator\(\)\(\)](#).

### 25.8.4 Member Data Documentation

#### 25.8.4.1 boost::gregorian::date airtsp::Date\_T::\_date

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

Referenced by [display\(\)](#), and [airtsp::store\\_date::operator\(\)\(\)](#).

#### **25.8.4.2 unsigned int airtsp::Date\_T::\_reldays**

Definition at line 30 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#).

#### **25.8.4.3 unsigned int airtsp::Date\_T::\_day**

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

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#), [display\(\)](#), and [getDate\(\)](#).

#### **25.8.4.4 unsigned int airtsp::Date\_T::\_month**

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

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#), [display\(\)](#), and [getDate\(\)](#).

#### **25.8.4.5 unsigned int airtsp::Date\_T::\_year**

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

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#), [display\(\)](#), and [getDate\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.hpp](#)

## **25.9 AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference**

```
#include <airtsp/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 > flight\\_period](#)
- [boost::spirit::classic::rule< ScannerT > not\\_to\\_be\\_parsed](#)
- [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 > [operating\\_leg\\_details](#)
- 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](#)

### 25.9.1 Detailed Description

**template<typename ScannerT> struct AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >**

Definition at line 287 of file [ScheduleParserHelper.hpp](#).

### 25.9.2 Constructor & Destructor Documentation

**25.9.2.1 template<typename ScannerT >  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition<  
ScannerT >::definition (FlightPeriodParser const & self) [inline]**

Definition at line 542 of file [ScheduleParserHelper.cpp](#).

References

AIRTSP::ScheduleParserHelper::FlightPeriodParser::_bomRoot,	
AIRTSP::FlightPeriodStruct::_dateOffset,	AIRTSP::ScheduleParserHelper::FlightPeriodParser::_-
flightPeriod,	AIRTSP::FlightPeriodStruct::_itDay,
AIRTSP::FlightPeriodStruct::_itHours,	AIRTSP::FlightPeriodStruct::_itMinutes,
AIRTSP::FlightPeriodStruct::_itMonth,	AIRTSP::FlightPeriodStruct::_itSeconds,
AIRTSP::FlightPeriodStruct::_itYear,	
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::airline_code,	
AIRTSP::ScheduleParserHelper::airline_code_p(),	AIRTSP::ScheduleParserHelper::airport_p(),
AIRTSP::ScheduleParserHelper::cabin_code_p(),	AIRTSP::ScheduleParserHelper::class_code_-
list_p(),	AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date,
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date_offset,	
AIRTSP::ScheduleParserHelper::day_p(),	AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::dow,
AIRTSP::ScheduleParserHelper::dow_p(),	AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::family_cabin_details,
AIRTSP::ScheduleParserHelper::family_-	
code_p,	AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_key,
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition<	

```
ScannerT          >::flight_number,          AIRTSP::ScheduleParserHelper::flight_number_p(),
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::flight_period,
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::flight_period_
end,          AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::flight_
period_list,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::full_
segment_cabin_details,          AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT          >::generic_segment,          AIRTSP::ScheduleParserHelper::hours_p(),
AIRTSP::ScheduleParserHelper::intl_p,          AIRTSP::ScheduleParserHelper::key_p(),
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::leg,
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::leg_cabin_
details,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::leg_
details,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::leg_
key,  AIRTSP::ScheduleParserHelper::minutes_p(),  AIRTSP::ScheduleParserHelper::month_
p(),  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT
>::not_to_be_parsed,          AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT          >::operating_leg_details,          AIRTSP::ScheduleParserHelper::seconds_p(),
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::segment_cabin_
details,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::segment_
key,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::segment_
section,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::specific_
segment_list,  AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT          >::time,
and AIRTSP::ScheduleParserHelper::year_p().
```

### 25.9.3 Member Function Documentation

**25.9.3.1** `template<typename ScannerT > bsc::rule< ScannerT > const &  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::start ()  
const [inline]`

Entry point of the parser.

Definition at line 701 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight\\_period\\_](#)  
[list](#).

### 25.9.4 Member Data Documentation

**25.9.4.1** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::flight_period_list`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#),  
and [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::start\(\)](#).

**25.9.4.2** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::flight_period`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.3** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::not_to_be_parsed`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.4** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::flight_period_end`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.5** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::flight_key`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.6** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::airline_code`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.7** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::flight_number`



Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.8** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.9** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::dow`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.10** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::time`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.11** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::date_offset`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.12** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.13** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::leg_key`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.14** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::operating_leg_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.15** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::leg_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.16** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::leg_cabin_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.17** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::segment_section`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.18** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::segment_key`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.19** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::full_segment_cabin_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.20** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::segment_cabin_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.21** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::full_family_cabin_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

**25.9.4.22** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::family_cabin_details`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.23** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::generic_segment`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

**25.9.4.24** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT  
>::specific_segment_list`

Definition at line 291 of file [ScheduleParserHelper.hpp](#).

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

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.10 airtsp::SearchStringParser::definition< ScannerT > Struct Template Reference

### Public Member Functions

- [definition](#) ([SearchStringParser](#) const &self)
- `boost::spirit::classic::rule< ScannerT > const & start () const`

### Public Attributes

- `boost::spirit::classic::rule< ScannerT > search\_string`
- `boost::spirit::classic::rule< ScannerT > places`
- `boost::spirit::classic::rule< ScannerT > place\_element`
- `boost::spirit::classic::rule< ScannerT > dates`
- `boost::spirit::classic::rule< ScannerT > date`
- `boost::spirit::classic::rule< ScannerT > month`
- `boost::spirit::classic::rule< ScannerT > day`
- `boost::spirit::classic::rule< ScannerT > year`
- `boost::spirit::classic::rule< ScannerT > preferred\_airlines`
- `boost::spirit::classic::rule< ScannerT > airline\_code`
- `boost::spirit::classic::rule< ScannerT > airline\_name`
- `boost::spirit::classic::rule< ScannerT > passengers`
- `boost::spirit::classic::rule< ScannerT > passenger\_number`
- `boost::spirit::classic::rule< ScannerT > passenger\_type`
- `boost::spirit::classic::rule< ScannerT > passenger\_adult\_type`
- `boost::spirit::classic::rule< ScannerT > passenger\_child\_type`
- `boost::spirit::classic::rule< ScannerT > passenger\_pet\_type`

### 25.10.1 Detailed Description

**template<typename ScannerT> struct airtsp::SearchStringParser::definition< ScannerT >**

Definition at line 259 of file [BookingRequestParser.cpp](#).

### 25.10.2 Constructor & Destructor Documentation

**25.10.2.1 template<typename ScannerT > airtsp::SearchStringParser::definition< ScannerT >::definition (SearchStringParser const & self) [inline]**

Definition at line 260 of file [BookingRequestParser.cpp](#).

References [airtsp::Date\\_T::\\_day](#), [airtsp::Date\\_T::\\_month](#), [airtsp::SearchStringParser::\\_searchString](#), [airtsp::SearchString\\_T::\\_tmpDate](#), [airtsp::Date\\_T::\\_year](#), [airtsp::SearchStringParser::definition< ScannerT >::airline\\_code](#), [airtsp::SearchStringParser::definition< ScannerT >::airline\\_name](#), [airtsp::SearchStringParser::definition< ScannerT >::date](#), [airtsp::SearchStringParser::definition< ScannerT >::dates](#), [airtsp::SearchStringParser::definition< ScannerT >::day](#), [airtsp::SearchStringParser::definition< ScannerT >::month](#), [airtsp::SearchStringParser::definition< ScannerT >::passenger\\_adult\\_type](#), [airtsp::SearchStringParser::definition< ScannerT >::passenger\\_child\\_type](#), [airtsp::SearchStringParser::definition< ScannerT >::passenger\\_number](#), [airtsp::SearchStringParser::definition< ScannerT >::passenger\\_pet\\_type](#), [airtsp::SearchStringParser::definition< ScannerT >::passenger\\_type](#), [airtsp::SearchStringParser::definition< ScannerT >::passengers](#), [airtsp::SearchStringParser::definition< ScannerT >::place\\_element](#), [airtsp::SearchStringParser::definition< ScannerT >::places](#), [airtsp::SearchStringParser::definition< ScannerT >::preferred\\_airlines](#), [airtsp::SearchStringParser::definition< ScannerT >::search\\_string](#), [airtsp::uint1\\_2\\_p](#), [airtsp::uint1\\_p](#), [airtsp::uint2\\_p](#), [airtsp::uint4\\_p](#), and [airtsp::SearchStringParser::definition< ScannerT >::year](#).

### 25.10.3 Member Function Documentation

**25.10.3.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> const& airtsp::SearchStringParser::definition< ScannerT >::start () const [inline]**

Definition at line 366 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchStringParser::definition< ScannerT >::search\\_string](#).

### 25.10.4 Member Data Documentation

**25.10.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> airtsp::SearchStringParser::definition< ScannerT >::search\_string**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#), and [airtsp::SearchStringParser::definition< ScannerT >::start\(\)](#).

**25.10.4.2    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::places**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.3    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::place\_element**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.4    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::dates**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.5    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::date**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.6    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::month**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.7    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::day**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.8    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::year**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.9    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::preferred\_airlines**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.10    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::airline\_code**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.11    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::airline\_name**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.12    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passengers**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.13    template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passenger\_number**

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.14** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passenger_type`

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.15** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passenger_adult_type`

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.16** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passenger_child_type`

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

**25.10.4.17** `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>  
airtsp::SearchStringParser::definition< ScannerT >::passenger_pet_type`

Definition at line 360 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

**25.11 AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT > Struct Template Reference**

```
#include <airtsp/command/OnDParserHelper.hpp>
```

**Public Member Functions**

- [definition](#) ([OnDParser](#) const &self)
- `boost::spirit::classic::rule< ScannerT > const & start () const`

**Public Attributes**

- `boost::spirit::classic::rule< ScannerT > ond\_list`
- `boost::spirit::classic::rule< ScannerT > ond`



- [boost::spirit::classic::rule< ScannerT > segment](#)
- [boost::spirit::classic::rule< ScannerT > ond\\_key](#)
- [boost::spirit::classic::rule< ScannerT > ond\\_end](#)
- [boost::spirit::classic::rule< ScannerT > date](#)
- [boost::spirit::classic::rule< ScannerT > time](#)

### 25.11.1 Detailed Description

**template<typename ScannerT> struct AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >**

Definition at line 133 of file [OnDParserHelper.hpp](#).

### 25.11.2 Constructor & Destructor Documentation

**25.11.2.1 template<typename ScannerT > AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition (OnDParser const & self) [inline]**

Definition at line 267 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDParserHelper::OnDParser::\\_bomRoot](#), [AIRTSP::OnDPeriodStruct::\\_itDay](#), [AIRTSP::OnDPeriodStruct::\\_itHours](#), [AIRTSP::OnDPeriodStruct::\\_itMinutes](#), [AIRTSP::OnDPeriodStruct::\\_itMonth](#), [AIRTSP::OnDPeriodStruct::\\_itSeconds](#), [AIRTSP::OnDPeriodStruct::\\_itYear](#), [AIRTSP::OnDParserHelper::OnDParser::\\_onDPeriod](#), [AIRTSP::OnDParserHelper::airline\\_code\\_p\(\)](#), [AIRTSP::OnDParserHelper::airport\\_p\(\)](#), [AIRTSP::OnDParserHelper::class\\_code\\_p\(\)](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::date](#), [AIRTSP::OnDParserHelper::day\\_p\(\)](#), [AIRTSP::OnDParserHelper::hours\\_p\(\)](#), [AIRTSP::OnDParserHelper::minutes\\_p\(\)](#), [AIRTSP::OnDParserHelper::month\\_p\(\)](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond\\_end](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond\\_key](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond\\_list](#), [AIRTSP::OnDParserHelper::seconds\\_p\(\)](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::segment](#), [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::time](#), and [AIRTSP::OnDParserHelper::year\\_p\(\)](#).

### 25.11.3 Member Function Documentation

**25.11.3.1 template<typename ScannerT > boost::spirit::classic::rule< ScannerT > const & AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::start () const [inline]**

Entry point of the parser.

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

References [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond\\_list](#).

#### 25.11.4 Member Data Documentation

##### 25.11.4.1 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond_list`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::start\(\)](#).

##### 25.11.4.2 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

##### 25.11.4.3 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::segment`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

##### 25.11.4.4 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond_key`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

##### 25.11.4.5 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::ond_end`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

##### 25.11.4.6 `template<typename ScannerT> boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::date`

Definition at line 137 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

#### 25.11.4.7 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::time`

Definition at line 137 of file [OnDParserHelper.hpp](#).

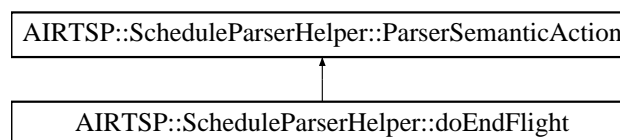
Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.12 AIRTSP::ScheduleParserHelper::doEndFlight Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::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](#)

#### 25.12.1 Detailed Description

Mark the end of the flight-period parsing.

Definition at line 224 of file [ScheduleParserHelper.hpp](#).

#### 25.12.2 Constructor & Destructor Documentation

##### 25.12.2.1 AIRTSP::ScheduleParserHelper::doEndFlight::doEndFlight (stdair::BomRoot & *ioBomRoot*, *FlightPeriodStruct* & *ioFlightPeriod*)

Actor Constructor.

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

### 25.12.3 Member Function Documentation

#### 25.12.3.1 void AIRTSP::ScheduleParserHelper::doEndFlight::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 448 of file [ScheduleParserHelper.cpp](#).

References [\\_bomRoot](#), [AIRTSP::LegStruct::\\_cabinList](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::FlightPeriodStruct::\\_legAlreadyDefined](#), [AIRTSP::FlightPeriodStruct::\\_legList](#), and [AIRTSP::FlightPeriodStruct::describe\(\)](#).

### 25.12.4 Member Data Documentation

#### 25.12.4.1 stdair::BomRoot& AIRTSP::ScheduleParserHelper::doEndFlight::\_bomRoot

Actor Specific Context.

Definition at line 230 of file [ScheduleParserHelper.hpp](#).

Referenced by [operator\(\)](#).

#### 25.12.4.2 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

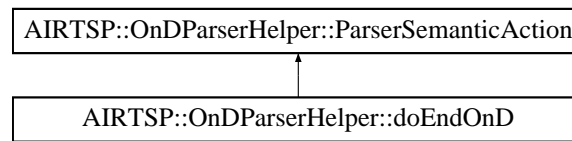
Referenced by [operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.13 AIRTSP::OnDParserHelper::doEndOnD Struct Reference

```
#include <airtsp/command/OnDParserHelper.hpp> Inheritance diagram for
AIRTSP::OnDParserHelper::doEndOnD::
```



### Public Member Functions

- [doEndOnD](#) (stdair::BomRoot &, [OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- stdair::BomRoot & [\\_bomRoot](#)
- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

#### 25.13.1 Detailed Description

Mark the end of the O&D parsing.

Definition at line 106 of file [OnDParserHelper.hpp](#).

#### 25.13.2 Constructor & Destructor Documentation

##### 25.13.2.1 AIRTSP::OnDParserHelper::doEndOnD::doEndOnD (stdair::BomRoot & *ioBomRoot*, OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 193 of file [OnDParserHelper.cpp](#).

#### 25.13.3 Member Function Documentation

##### 25.13.3.1 void AIRTSP::OnDParserHelper::doEndOnD::operator() ([iterator\\_t](#) *iStr*, [iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 199 of file [OnDParserHelper.cpp](#).

References [\\_bomRoot](#), and [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#).

#### 25.13.4 Member Data Documentation

##### 25.13.4.1 stdair::BomRoot& AIRTSP::OnDParserHelper::doEndOnD::\_bomRoot

Actor Specific Context.

Definition at line 112 of file [OnDParserHelper.hpp](#).

Referenced by [operator\(\)](#).

#### 25.13.4.2 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

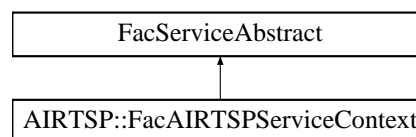
The documentation for this struct was generated from the following files:

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.14 AIRTSP::FacAIRTSPServiceContext Class Reference

Factory for the service context.

`#include <airtsp/factory/FacAIRTSPServiceContext.hpp>`Inheritance diagram for AIRTSP::FacAIRTSPServiceContext::



### Public Member Functions

- [~FacAIRTSPServiceContext\(\)](#)
- [AIRTSP\\_ServiceContext & create\(\)](#)

### Static Public Member Functions

- static [FacAIRTSPServiceContext & instance\(\)](#)

### Protected Member Functions

- [FacAIRTSPServiceContext\(\)](#)

#### 25.14.1 Detailed Description

Factory for the service context.

Definition at line 19 of file [FacAIRTSPServiceContext.hpp](#).

## 25.14.2 Constructor & Destructor Documentation

### 25.14.2.1 AIRTSP::FacAIRTSPServiceContext::~~FacAIRTSPServiceContext ()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacAIRTSPServiceContext::instance\(\)](#).

Definition at line 17 of file [FacAIRTSPServiceContext.cpp](#).

### 25.14.2.2 AIRTSP::FacAIRTSPServiceContext::FacAIRTSPServiceContext () [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

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

Referenced by [instance\(\)](#).

## 25.14.3 Member Function Documentation

### 25.14.3.1 FacAIRTSPServiceContext & AIRTSP::FacAIRTSPServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

#### Returns:

FacServiceContext&

Definition at line 22 of file [FacAIRTSPServiceContext.cpp](#).

References [FacAIRTSPServiceContext\(\)](#).

### 25.14.3.2 AIRTSP\_ServiceContext & AIRTSP::FacAIRTSPServiceContext::create ()

Create a new ServiceContext object.

This new object is added to the list of instantiated objects.

#### Returns:

ServiceContext& The newly created object.

Definition at line 34 of file [FacAIRTSPServiceContext.cpp](#).

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

- [airtsp/factory/FacAIRTSPServiceContext.hpp](#)
- [airtsp/factory/FacAIRTSPServiceContext.cpp](#)

## 25.15 AIRTSP::FacServiceAbstract Class Reference

```
#include <airtsp/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](#)

#### 25.15.1 Detailed Description

Base class for the (Service) Factory layer.

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

#### 25.15.2 Member Typedef Documentation

##### 25.15.2.1 typedef std::vector<ServiceAbstract\*> AIRTSP::FacServiceAbstract::ServicePool\_T

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

#### 25.15.3 Constructor & Destructor Documentation

##### 25.15.3.1 AIRTSP::FacServiceAbstract::~~FacServiceAbstract () [virtual]

Destructor.

Definition at line 13 of file [FacServiceAbstract.cpp](#).

References [clean\(\)](#).

##### 25.15.3.2 AIRTSP::FacServiceAbstract::FacServiceAbstract () [inline, protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

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



### 25.15.4 Member Function Documentation

#### 25.15.4.1 void AIRTSP::FacServiceAbstract::clean ()

Destroyed all the object instantiated by this factory.

Definition at line 18 of file [FacServiceAbstract.cpp](#).

References [\\_pool](#).

Referenced by [~FacServiceAbstract\(\)](#).

### 25.15.5 Member Data Documentation

#### 25.15.5.1 ServicePool\_T AIRTSP::FacServiceAbstract::\_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file [FacServiceAbstract.hpp](#).

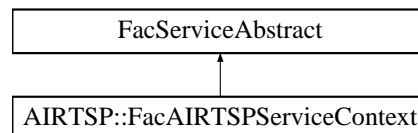
Referenced by [clean\(\)](#).

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

- [airtsp/factory/FacServiceAbstract.hpp](#)
- [airtsp/factory/FacServiceAbstract.cpp](#)

## 25.16 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract::



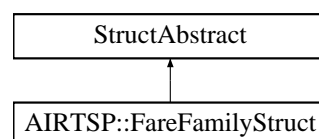
The documentation for this class was generated from the following file:

- [airtsp/factory/FacAIRTSPServiceContext.hpp](#)

## 25.17 AIRTSP::FareFamilyStruct Struct Reference

`#include <airtsp/bom/FareFamilyStruct.hpp>`  
 AIRTSP::FareFamilyStruct::

diagram for



### Public Member Functions

- [FareFamilyStruct](#) (const stdair::FamilyCode\_T &, const stdair::CurveKey\_T &, const stdair::CurveKey\_T &, const stdair::ClassList\_String\_T &)
- const std::string [describe](#) () const

### Public Attributes

- stdair::FamilyCode\_T [\\_familyCode](#)
- stdair::CurveKey\_T [\\_frat5CurveKey](#)
- stdair::CurveKey\_T [\\_ffDisutilityCurveKey](#)
- stdair::ClassList\_String\_T [\\_classes](#)

### 25.17.1 Detailed Description

Utility Structure for the parsing of fare family details.

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

### 25.17.2 Constructor & Destructor Documentation

- 25.17.2.1 AIRTSP::FareFamilyStruct::FareFamilyStruct (const stdair::FamilyCode\_T & *iFamilyCode*, const stdair::CurveKey\_T & *iFRAT5Key*, const stdair::CurveKey\_T & *iFFDisutilityKey*, const stdair::ClassList\_String\_T & *iClasses*)**

Constructors.

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

### 25.17.3 Member Function Documentation

- 25.17.3.1 const std::string AIRTSP::FareFamilyStruct::describe () const**

Give a description of the structure (for display purposes).

Definition at line 23 of file [FareFamilyStruct.cpp](#).

References [\\_classes](#), [\\_familyCode](#), [\\_ffDisutilityCurveKey](#), and [\\_frat5CurveKey](#).

### 25.17.4 Member Data Documentation

- 25.17.4.1 stdair::FamilyCode\_T AIRTSP::FareFamilyStruct::\_familyCode**

Definition at line 19 of file [FareFamilyStruct.hpp](#).

Referenced by [describe\(\)](#).

- 25.17.4.2 stdair::CurveKey\_T AIRTSP::FareFamilyStruct::\_frat5CurveKey**

Definition at line 20 of file [FareFamilyStruct.hpp](#).

Referenced by [describe\(\)](#).

#### 25.17.4.3 stdair::CurveKey\_T AIRTSP::FareFamilyStruct::\_ffDisutilityCurveKey

Definition at line 21 of file [FareFamilyStruct.hpp](#).

Referenced by [describe\(\)](#).

#### 25.17.4.4 stdair::ClassList\_String\_T AIRTSP::FareFamilyStruct::\_classes

Definition at line 22 of file [FareFamilyStruct.hpp](#).

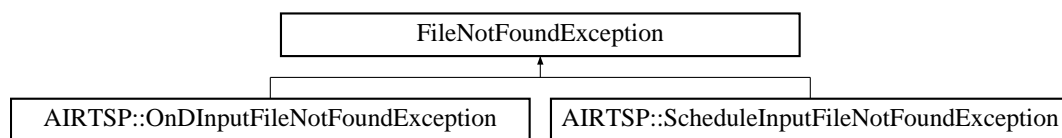
Referenced by [describe\(\)](#).

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

- [airtsp/bom/FareFamilyStruct.hpp](#)
- [airtsp/bom/FareFamilyStruct.cpp](#)

## 25.18 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException::



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

- [airtsp/AIRTSP\\_Types.hpp](#)

## 25.19 AIRTSP::FlagSaver Struct Reference

### Public Member Functions

- [FlagSaver](#) (std::ostream &oStream)
- [~FlagSaver](#) ()

#### 25.19.1 Detailed Description

Helper singleton structure to store the current formatting flags of any given output stream. The flags are re-set at the structure destruction.

Definition at line 22 of file [BomDisplay.cpp](#).

## 25.19.2 Constructor & Destructor Documentation

### 25.19.2.1 AIRTSP::FlagSaver::FlagSaver (std::ostream & oStream) [inline]

Constructor.

Definition at line 25 of file [BomDisplay.cpp](#).

### 25.19.2.2 AIRTSP::FlagSaver::~~FlagSaver () [inline]

Destructor.

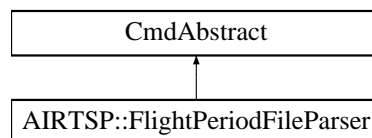
Definition at line 30 of file [BomDisplay.cpp](#).

The documentation for this struct was generated from the following file:

- [airtsp/bom/BomDisplay.cpp](#)

## 25.20 AIRTSP::FlightPeriodFileParser Class Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::FlightPeriodFileParser::



### Public Member Functions

- [FlightPeriodFileParser](#) (stdair::BomRoot &ioBomRoot, const stdair::Filename\_T &iFilename)
- bool [generateInventories](#) ()

### 25.20.1 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 323 of file [ScheduleParserHelper.hpp](#).

### 25.20.2 Constructor & Destructor Documentation

#### 25.20.2.1 AIRTSP::FlightPeriodFileParser::FlightPeriodFileParser (stdair::BomRoot & ioBomRoot, const stdair::Filename\_T &iFilename)

Constructor.

Definition at line 716 of file [ScheduleParserHelper.cpp](#).

### 25.20.3 Member Function Documentation

#### 25.20.3.1 bool AIRTSP::FlightPeriodFileParser::generateInventories ()

Parse the input file and generate the Inventories.

Definition at line 753 of file [ScheduleParserHelper.cpp](#).

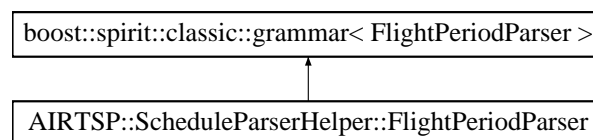
Referenced by [AIRTSP::ScheduleParser::generateInventories\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.21 AIRTSP::ScheduleParserHelper::FlightPeriodParser Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::FlightPeriodParser::



### Classes

- struct [definition](#)

### Public Member Functions

- [FlightPeriodParser](#) (stdair::BomRoot &, [FlightPeriodStruct](#) &)

### Public Attributes

- stdair::BomRoot & [\\_bomRoot](#)
- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.21.1 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 ';' End-OfFlight EndOfFlight ::= ';' Grammar for the Flight-Period parser.

Definition at line 281 of file [ScheduleParserHelper.hpp](#).

### 25.21.2 Constructor & Destructor Documentation

#### 25.21.2.1 AIRTSP::ScheduleParserHelper::FlightPeriodParser::FlightPeriodParser (stdair::BomRoot & ioBomRoot, FlightPeriodStruct & ioFlightPeriod)

Definition at line 533 of file [ScheduleParserHelper.cpp](#).

### 25.21.3 Member Data Documentation

#### 25.21.3.1 stdair::BomRoot& AIRTSP::ScheduleParserHelper::FlightPeriodParser::\_bomRoot

Definition at line 304 of file [ScheduleParserHelper.hpp](#).

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

#### 25.21.3.2 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::FlightPeriodParser::\_flightPeriod

Definition at line 305 of file [ScheduleParserHelper.hpp](#).

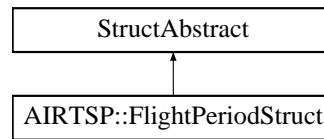
Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.22 AIRTSP::FlightPeriodStruct Struct Reference

`#include <airtsp/bom/FlightPeriodStruct.hpp>` Inheritance diagram for AIRTSP::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](#)

#### 25.22.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

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

### 25.22.2 Constructor & Destructor Documentation

#### 25.22.2.1 AIRTSP::FlightPeriodStruct::FlightPeriodStruct ()

Constructor.

Definition at line 17 of file [FlightPeriodStruct.cpp](#).

### 25.22.3 Member Function Documentation

#### 25.22.3.1 stdair::Date\_T AIRTSP::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 [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.22.3.2 stdair::Duration\_T AIRTSP::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 [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#).

#### 25.22.3.3 const std::string AIRTSP::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](#), [AIRTSP::SegmentStruct::describe\(\)](#), and [AIRTSP::LegStruct::describe\(\)](#).

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#).

#### 25.22.3.4 void AIRTSP::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 [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#).

#### 25.22.3.5 void AIRTSP::FlightPeriodStruct::buildSegments ()

Build the list of [SegmentStruct](#) objects.



Definition at line 78 of file [FlightPeriodStruct.cpp](#).

References [\\_airportList](#), [\\_airportOrderedList](#), [AIRTSP::SegmentStruct::\\_boardingPoint](#), [AIRTSP::SegmentStruct::\\_offPoint](#), and [\\_segmentList](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

### 25.22.3.6 void AIRTSP::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 [AIRTSP::SegmentStruct::\\_boardingPoint](#), [AIRTSP::SegmentStruct::\\_offPoint](#), and [\\_segmentList](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#).

### 25.22.3.7 void AIRTSP::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 149 of file [FlightPeriodStruct.cpp](#).

References [AIRTSP::SegmentStruct::\\_cabinList](#), and [\\_segmentList](#).

### 25.22.3.8 void AIRTSP::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 162 of file [FlightPeriodStruct.cpp](#).

References [AIRTSP::SegmentStruct::\\_boardingPoint](#), [AIRTSP::SegmentCabinStruct::\\_cabinCode](#), [AIRTSP::SegmentStruct::\\_offPoint](#), and [\\_segmentList](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)](#).

### 25.22.3.9 void AIRTSP::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 229 of file [FlightPeriodStruct.cpp](#).

References [AIRTSP::SegmentCabinStruct::\\_cabinCode](#), [AIRTSP::SegmentStruct::\\_cabinList](#), and [\\_segmentList](#).

#### 25.22.4 Member Data Documentation

##### 25.22.4.1 stdair::AirlineCode\_T AIRTSP::FlightPeriodStruct::\_airlineCode

Definition at line 84 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

##### 25.22.4.2 stdair::FlightNumber\_T AIRTSP::FlightPeriodStruct::\_flightNumber

Definition at line 85 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#).

##### 25.22.4.3 stdair::DatePeriod\_T AIRTSP::FlightPeriodStruct::\_dateRange

Definition at line 86 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

##### 25.22.4.4 stdair::DoWStruct AIRTSP::FlightPeriodStruct::\_dow

Definition at line 87 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#).

##### 25.22.4.5 LegStructList\_T AIRTSP::FlightPeriodStruct::\_legList

Definition at line 88 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

##### 25.22.4.6 SegmentStructList\_T AIRTSP::FlightPeriodStruct::\_segmentList

Definition at line 89 of file [FlightPeriodStruct.hpp](#).

Referenced by [addFareFamily\(\)](#), [addSegmentCabin\(\)](#), [buildSegments\(\)](#), and [describe\(\)](#).

#### 25.22.4.7 bool AIRTSP::FlightPeriodStruct::\_legAlreadyDefined

Staging Leg (resp. Cabin) structure, gathering the result of the iteration on one leg (resp. cabin).

Definition at line 93 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

#### 25.22.4.8 LegStruct AIRTSP::FlightPeriodStruct::\_itLeg

Definition at line 94 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.22.4.9 LegCabinStruct AIRTSP::FlightPeriodStruct::\_itLegCabin

Definition at line 95 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#).

#### 25.22.4.10 std::date T AIRTSP::FlightPeriodStruct::\_dateRangeStart

Staging Date.

Definition at line 98 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#).

#### 25.22.4.11 std::date T AIRTSP::FlightPeriodStruct::\_dateRangeEnd

Definition at line 99 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.22.4.12 unsigned int AIRTSP::FlightPeriodStruct::\_itYear

Definition at line 100 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.22.4.13 unsigned int AIRTSP::FlightPeriodStruct::\_itMonth

Definition at line 101 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.22.4.14 unsigned int AIRTSP::FlightPeriodStruct::\_itDay

Definition at line 102 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.22.4.15 int AIRTSP::FlightPeriodStruct::\_dateOffset

Definition at line 103 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#).

#### 25.22.4.16 long AIRTSP::FlightPeriodStruct::\_itHours

Staging Time.

Definition at line 106 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), and [getTime\(\)](#).

#### 25.22.4.17 long AIRTSP::FlightPeriodStruct::\_itMinutes

Definition at line 107 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), and [getTime\(\)](#).

#### 25.22.4.18 long AIRTSP::FlightPeriodStruct::\_itSeconds

Definition at line 108 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#), [getTime\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#).

#### 25.22.4.19 AirportList\_T AIRTSP::FlightPeriodStruct::\_airportList

Staging Airport List (helper to derive the list of Segment structures).

Definition at line 112 of file [FlightPeriodStruct.hpp](#).

Referenced by [addAirport\(\)](#), [buildSegments\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

#### 25.22.4.20 AirportOrderedList\_T AIRTSP::FlightPeriodStruct::\_airportOrderedList

Definition at line 113 of file [FlightPeriodStruct.hpp](#).

Referenced by [addAirport\(\)](#), [buildSegments\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

#### 25.22.4.21 bool AIRTSP::FlightPeriodStruct::\_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 116 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

#### 25.22.4.22 SegmentStruct AIRTSP::FlightPeriodStruct::\_itSegment

Definition at line 117 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

#### 25.22.4.23 SegmentCabinStruct AIRTSP::FlightPeriodStruct::\_itSegmentCabin

Definition at line 118 of file [FlightPeriodStruct.hpp](#).

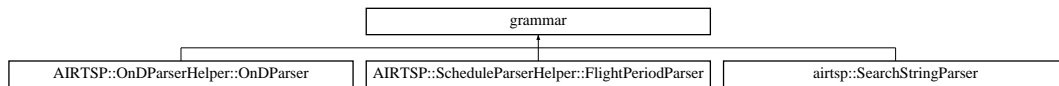
Referenced by [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#).

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

- [airtsp/bom/FlightPeriodStruct.hpp](#)
- [airtsp/bom/FlightPeriodStruct.cpp](#)

## 25.23 grammar Class Reference

Inheritance diagram for grammar::

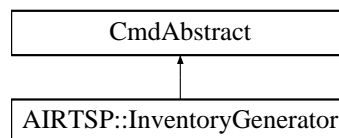


The documentation for this class was generated from the following files:

- airtsp/batches/[BookingRequestParser.cpp](#)
- airtsp/command/[ScheduleParserHelper.hpp](#)
- airtsp/command/[OnDParserHelper.hpp](#)

## 25.24 AIRTSP::InventoryGenerator Class Reference

`#include <airtsp/command/InventoryGenerator.hpp>`  
 Inheritance diagram for AIRTSP::InventoryGenerator::



### Friends

- class [FlightPeriodFileParser](#)
- class [FFFlightPeriodFileParser](#)
- struct [ScheduleParserHelper::doEndFlight](#)
- class [ScheduleParser](#)

### 25.24.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

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

### 25.24.2 Friends And Related Function Documentation

#### 25.24.2.1 friend class FlightPeriodFileParser [friend]

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

**25.24.2.2 friend class FFFlightPeriodFileParser [friend]**

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

**25.24.2.3 friend struct ScheduleParserHelper::doEndFlight [friend]**

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

**25.24.2.4 friend class ScheduleParser [friend]**

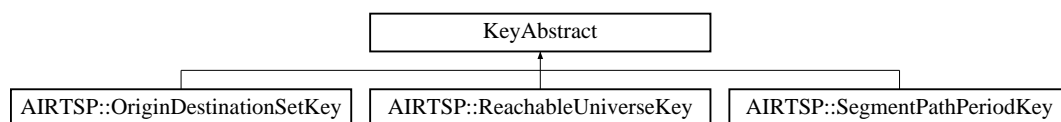
Definition at line 38 of file [InventoryGenerator.hpp](#).

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

- [airtsp/command/InventoryGenerator.hpp](#)
- [airtsp/command/InventoryGenerator.cpp](#)

**25.25 KeyAbstract Class Reference**

Inheritance diagram for KeyAbstract::



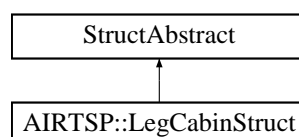
The documentation for this class was generated from the following files:

- [airtsp/bom/OriginDestinationSetKey.hpp](#)
- [airtsp/bom/ReachableUniverseKey.hpp](#)
- [airtsp/bom/SegmentPathPeriodKey.hpp](#)

**25.26 AIRTSP::LegCabinStruct Struct Reference**

`#include <airtsp/bom/LegCabinStruct.hpp>`  
 AIRTSP::LegCabinStruct::

diagram for



### Public Member Functions

- void [fill](#) (stdair::LegCabin &) const
- const std::string [describe](#) () const

### Public Attributes

- stdair::CabinCode\_T [\\_cabinCode](#)
- stdair::CabinCapacity\_T [\\_capacity](#)

#### 25.26.1 Detailed Description

Utility Structure for the parsing of LegCabin details.

Definition at line 22 of file [LegCabinStruct.hpp](#).

#### 25.26.2 Member Function Documentation

##### 25.26.2.1 void AIRTSP::LegCabinStruct::fill (stdair::LegCabin & *ioLegCabin*) const

Fill the LegCabin objects with the attributes of the [LegCabinStruct](#).

Definition at line 22 of file [LegCabinStruct.cpp](#).

References [\\_capacity](#).

##### 25.26.2.2 const std::string AIRTSP::LegCabinStruct::describe () const

Give a description of the structure (for display purposes).

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

References [\\_cabinCode](#), and [\\_capacity](#).

Referenced by [AIRTSP::LegStruct::describe\(\)](#).

#### 25.26.3 Member Data Documentation

##### 25.26.3.1 stdair::CabinCode\_T AIRTSP::LegCabinStruct::\_cabinCode

Definition at line 24 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#).

##### 25.26.3.2 stdair::CabinCapacity\_T AIRTSP::LegCabinStruct::\_capacity

Definition at line 25 of file [LegCabinStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#).

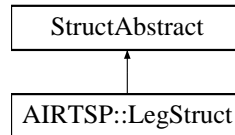
The documentation for this struct was generated from the following files:



- [airtsp/bom/LegCabinStruct.hpp](#)
- [airtsp/bom/LegCabinStruct.cpp](#)

## 25.27 AIRTSP::LegStruct Struct Reference

`#include <airtsp/bom/LegStruct.hpp>`Inheritance diagram for AIRTSP::LegStruct:



### Public Member Functions

- void [fill](#) (const stdair::Date\_T &iRefDate, stdair::LegDate &) const
- const std::string [describe](#) () const
- [LegStruct](#) ()

### Public Attributes

- stdair::AirlineCode\_T [\\_airlineCode](#)
- stdair::FlightNumber\_T [\\_flightNumber](#)
- stdair::AirportCode\_T [\\_boardingPoint](#)
- stdair::DateOffset\_T [\\_boardingDateOffset](#)
- stdair::Duration\_T [\\_boardingTime](#)
- stdair::AirportCode\_T [\\_offPoint](#)
- stdair::DateOffset\_T [\\_offDateOffset](#)
- stdair::Duration\_T [\\_offTime](#)
- stdair::Duration\_T [\\_elapsed](#)
- [LegCabinStructList\\_T](#) [\\_cabinList](#)

### 25.27.1 Detailed Description

Utility Structure for the parsing of Leg structures.

Definition at line 24 of file [LegStruct.hpp](#).

### 25.27.2 Constructor & Destructor Documentation

#### 25.27.2.1 AIRTSP::LegStruct::LegStruct ()

Default Constructor.

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

### 25.27.3 Member Function Documentation

#### 25.27.3.1 void AIRTSP::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 48 of file [LegStruct.cpp](#).

References [\\_airlineCode](#), [\\_boardingDateOffset](#), [\\_boardingTime](#), [\\_elapsed](#), [\\_flightNumber](#), [\\_offDateOffset](#), [\\_offPoint](#), and [\\_offTime](#).

#### 25.27.3.2 const std::string AIRTSP::LegStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 22 of file [LegStruct.cpp](#).

References [\\_boardingDateOffset](#), [\\_boardingPoint](#), [\\_boardingTime](#), [\\_cabinList](#), [\\_elapsed](#), [\\_offDateOffset](#), [\\_offPoint](#), [\\_offTime](#), and [AIRTSP::LegCabinStruct::describe\(\)](#).

Referenced by [AIRTSP::FlightPeriodStruct::describe\(\)](#).

### 25.27.4 Member Data Documentation

#### 25.27.4.1 stdair::AirlineCode\_T AIRTSP::LegStruct::\_airlineCode

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

Referenced by [fill\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.27.4.2 stdair::FlightNumber\_T AIRTSP::LegStruct::\_flightNumber

Definition at line 27 of file [LegStruct.hpp](#).

Referenced by [fill\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.27.4.3 stdair::AirportCode\_T AIRTSP::LegStruct::\_boardingPoint

Definition at line 28 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRTSP::SegmentPeriodHelper::fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

**25.27.4.4 stdair::DateOffset\_T AIRTSP::LegStruct::\_boardingDateOffset**

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

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#).

**25.27.4.5 stdair::Duration\_T AIRTSP::LegStruct::\_boardingTime**

Definition at line 30 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#).

**25.27.4.6 stdair::AirportCode\_T AIRTSP::LegStruct::\_offPoint**

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

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#).

**25.27.4.7 stdair::DateOffset\_T AIRTSP::LegStruct::\_offDateOffset**

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

Referenced by [describe\(\)](#), [AIRTSP::SegmentPeriodHelper::fill\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

**25.27.4.8 stdair::Duration\_T AIRTSP::LegStruct::\_offTime**

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

Referenced by [describe\(\)](#), [AIRTSP::SegmentPeriodHelper::fill\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#).

**25.27.4.9 stdair::Duration\_T AIRTSP::LegStruct::\_elapsed**

Definition at line 34 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

**25.27.4.10 LegCabinStructList\_T AIRTSP::LegStruct::\_cabinList**

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

Referenced by [describe\(\)](#), [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

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

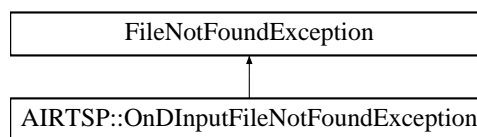
- [airtsp/bom/LegStruct.hpp](#)
- [airtsp/bom/LegStruct.cpp](#)

## 25.28 AIRTSP::OnDInputFileNotFoundException Class Reference

`#include <airtsp/AIRTSP_Types.hpp>`  
 Inheritance diagram for AIRTSP::OnDInputFileNotFoundException:

diagram

for



### Public Member Functions

- [OnDInputFileNotFoundException](#) (const std::string &iWhat)

### 25.28.1 Detailed Description

The O&D input file cannot be retrieved.

Definition at line 35 of file [AIRTSP\\_Types.hpp](#).

### 25.28.2 Constructor & Destructor Documentation

#### 25.28.2.1 AIRTSP::OnDInputFileNotFoundException::OnDInputFileNotFoundException (const std::string &iWhat) [inline]

Constructor.

Definition at line 40 of file [AIRTSP\\_Types.hpp](#).

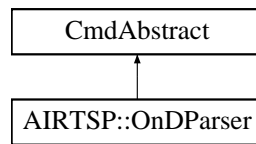
The documentation for this class was generated from the following file:

- [airtsp/AIRTSP\\_Types.hpp](#)

## 25.29 AIRTSP::OnDParser Class Reference

Class wrapping the parser entry point.

`#include <airtsp/command/OnDParser.hpp>`  
 Inheritance diagram for AIRTSP::OnDParser::



### Static Public Member Functions

- static void [generateOnDPeriods](#) (const stdair::ODFilePath &, stdair::BomRoot &)

#### 25.29.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 24 of file [OnDParser.hpp](#).

#### 25.29.2 Member Function Documentation

##### 25.29.2.1 void AIRTSP::OnDParser::generateOnDPeriods (const stdair::ODFilePath & iODFilename, stdair::BomRoot & ioBomRoot) [static]

Parse the CSV file describing the O&D.

#### Parameters:

**const** stdair::ODFilePath& The file-name of the CSV-formatted fare input file and the container.

Definition at line 17 of file [OnDParser.cpp](#).

References [AIRTSP::OnDPeriodFileParser::generateOnDPeriods\(\)](#).

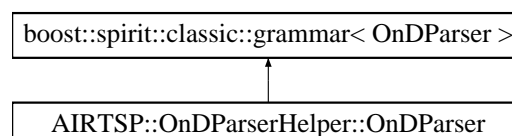
Referenced by [AIRTSP::AIRTSP\\_Service::parseAndLoad\(\)](#).

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

- [airtsp/command/OnDParser.hpp](#)
- [airtsp/command/OnDParser.cpp](#)

## 25.30 AIRTSP::OnDParserHelper::OnDParser Struct Reference

#include <airtsp/command/OnDParserHelper.hpp> Inheritance diagram for AIRTSP::OnDParserHelper::OnDParser::



### Classes

- struct [definition](#)

**Public Member Functions**

- [OnDParser](#) (stdair::BomRoot &, [OnDPeriodStruct](#) &)

**Public Attributes**

- stdair::BomRoot & [\\_bomRoot](#)
- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

**25.30.1 Detailed Description**

Fares: AirlineCode; OriginCity; DestinationCity; DepartureDate-Range(FirstDate; LastDate); Airline; Class; BA; NCE; LHR; 2007-01-01; 2007-12-31; BA; Y; BA; Y BA; NCE; LHR; 2007-01-01; 2007-12-31; BA; V; BA; H Grammar for the FareRule parser.

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

**25.30.2 Constructor & Destructor Documentation****25.30.2.1 AIRTSP::OnDParserHelper::OnDParser::OnDParser (stdair::BomRoot & *ioBomRoot*, OnDPeriodStruct & *ioOnDPeriod*)**

Definition at line 261 of file [OnDParserHelper.cpp](#).

**25.30.3 Member Data Documentation****25.30.3.1 stdair::BomRoot& AIRTSP::OnDParserHelper::OnDParser::\_bomRoot**

Definition at line 145 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

**25.30.3.2 OnDPeriodStruct& AIRTSP::OnDParserHelper::OnDParser::\_onDPeriod**

Definition at line 146 of file [OnDParserHelper.hpp](#).

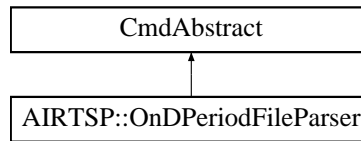
Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

**25.31 AIRTSP::OnDPeriodFileParser Class Reference**

`#include <airtsp/command/OnDParserHelper.hpp>` Inheritance diagram for AIRTSP::OnDPeriodFileParser::



### Public Member Functions

- [OnDPeriodFileParser](#) (const stdair::Filename\_T &iFilename, stdair::BomRoot &ioBomRoot)
- bool [generateOnDPeriods](#) ()

#### 25.31.1 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 161 of file [OnDParserHelper.hpp](#).

#### 25.31.2 Constructor & Destructor Documentation

##### 25.31.2.1 AIRTSP::OnDPeriodFileParser::OnDPeriodFileParser (const stdair::Filename\_T &iFilename, stdair::BomRoot &ioBomRoot)

Constructor.

Definition at line 342 of file [OnDParserHelper.cpp](#).

#### 25.31.3 Member Function Documentation

##### 25.31.3.1 bool AIRTSP::OnDPeriodFileParser::generateOnDPeriods ()

Parse the input file and generate the O&D-Periods.

Definition at line 378 of file [OnDParserHelper.cpp](#).

Referenced by [AIRTSP::OnDParser::generateOnDPeriods\(\)](#).

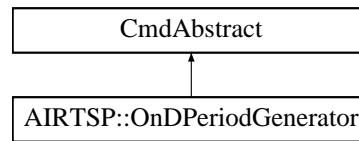
The documentation for this class was generated from the following files:

- airtsp/command/[OnDParserHelper.hpp](#)
- airtsp/command/[OnDParserHelper.cpp](#)

## 25.32 AIRTSP::OnDPeriodGenerator Class Reference

Class handling the generation / instantiation of the O&D-Period BOM.

#include <airtsp/command/OnDPeriodGenerator.hpp> Inheritance diagram for AIRTSP::OnDPeriodGenerator::



### Friends

- class [OnDPeriodFileParser](#)
- struct [OnDParserHelper::doEndOnD](#)
- class [OnDParser](#)

#### 25.32.1 Detailed Description

Class handling the generation / instantiation of the O&D-Period BOM.

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

#### 25.32.2 Friends And Related Function Documentation

##### 25.32.2.1 friend class OnDPeriodFileParser [friend]

Only the following class may use methods of [OnDPeriodGenerator](#). Indeed, as those methods build the BOM, it is not good to expose them publicly.

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

##### 25.32.2.2 friend struct OnDParserHelper::doEndOnD [friend]

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

##### 25.32.2.3 friend class OnDParser [friend]

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

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

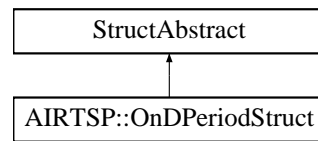
- [airtsp/command/OnDPeriodGenerator.hpp](#)
- [airtsp/command/OnDPeriodGenerator.cpp](#)

### 25.33 AIRTSP::OnDPeriodStruct Struct Reference

`#include <airtsp/bom/OnDPeriodStruct.hpp>`  
 AIRTSP::OnDPeriodStruct::

diagram for





### Public Member Functions

- const stdair::AirlineCode\_T & [getFirstAirlineCode](#) () const
- stdair::Date\_T [getDate](#) () const
- stdair::Duration\_T [getTime](#) () const
- const std::string [describe](#) () const
- const std::string [describeTSKey](#) () const
- [OnDPeriodStruct](#) ()

### Public Attributes

- stdair::AirportCode\_T [\\_origin](#)
- stdair::AirportCode\_T [\\_destination](#)
- stdair::DatePeriod\_T [\\_datePeriod](#)
- stdair::Duration\_T [\\_timeRangeStart](#)
- stdair::Duration\_T [\\_timeRangeEnd](#)
- stdair::NbOfAirlines\_T [\\_nbOfAirlines](#)
- stdair::AirlineCode\_T [\\_airlineCode](#)
- stdair::ClassCode\_T [\\_classCode](#)
- stdair::AirlineCodeList\_T [\\_airlineCodeList](#)
- stdair::ClassCodeList\_T [\\_classCodeList](#)
- stdair::Date\_T [\\_dateRangeStart](#)
- stdair::Date\_T [\\_dateRangeEnd](#)
- unsigned int [\\_itYear](#)
- unsigned int [\\_itMonth](#)
- unsigned int [\\_itDay](#)
- long [\\_itHours](#)
- long [\\_itMinutes](#)
- long [\\_itSeconds](#)

#### 25.33.1 Detailed Description

Utility Structure for the parsing of FareRule structures.

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

#### 25.33.2 Constructor & Destructor Documentation

##### 25.33.2.1 AIRTSP::OnDPeriodStruct::OnDPeriodStruct ()

Default constructor.

Definition at line 17 of file [OnDPeriodStruct.cpp](#).

### 25.33.3 Member Function Documentation

#### 25.33.3.1 const stdair::AirlineCode\_T & AIRTSP::OnDPeriodStruct::getFirstAirlineCode () const

Get the first airline code.

Definition at line 64 of file [OnDPeriodStruct.cpp](#).

References [\\_airlineCodeList](#).

#### 25.33.3.2 stdair::Date\_T AIRTSP::OnDPeriodStruct::getDate () const

Get the date from the staging details.

Definition at line 28 of file [OnDPeriodStruct.cpp](#).

References [\\_itDay](#), [\\_itMonth](#), and [\\_itYear](#).

Referenced by [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#).

#### 25.33.3.3 stdair::Duration\_T AIRTSP::OnDPeriodStruct::getTime () const

Get the time from the staging details.

Definition at line 33 of file [OnDPeriodStruct.cpp](#).

References [\\_itHours](#), [\\_itMinutes](#), and [\\_itSeconds](#).

Referenced by [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#).

#### 25.33.3.4 const std::string AIRTSP::OnDPeriodStruct::describe () const

Give a description of the structure (for display purposes).

Definition at line 40 of file [OnDPeriodStruct.cpp](#).

References [\\_airlineCode](#), [\\_classCode](#), [\\_datePeriod](#), [\\_destination](#), [\\_origin](#), [\\_timeRangeEnd](#), and [\\_timeRangeStart](#).

#### 25.33.3.5 const std::string AIRTSP::OnDPeriodStruct::describeTSKey () const

Give a short description of the key required in the travel solution object to differentiate fare rule structures.

Definition at line 55 of file [OnDPeriodStruct.cpp](#).

References [\\_airlineCode](#), [\\_classCode](#), [\\_destination](#), and [\\_origin](#).

### 25.33.4 Member Data Documentation

#### 25.33.4.1 stdair::AirportCode\_T AIRTSP::OnDPeriodStruct::\_origin

Definition at line 42 of file [OnDPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

#### 25.33.4.2 stdair::AirportCode\_T AIRTSP::OnDPeriodStruct::\_destination

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

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), and [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#).

#### 25.33.4.3 stdair::DatePeriod\_T AIRTSP::OnDPeriodStruct::\_datePeriod

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

Referenced by [describe\(\)](#), and [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#).

#### 25.33.4.4 stdair::Duration\_T AIRTSP::OnDPeriodStruct::\_timeRangeStart

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

Referenced by [describe\(\)](#), and [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#).

#### 25.33.4.5 stdair::Duration\_T AIRTSP::OnDPeriodStruct::\_timeRangeEnd

Definition at line 46 of file [OnDPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#).

#### 25.33.4.6 stdair::NbOfAirlines\_T AIRTSP::OnDPeriodStruct::\_nbOfAirlines

Definition at line 47 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

#### 25.33.4.7 stdair::AirlineCode\_T AIRTSP::OnDPeriodStruct::\_airlineCode

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

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

#### 25.33.4.8 stdair::ClassCode\_T AIRTSP::OnDPeriodStruct::\_classCode

Definition at line 49 of file [OnDPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#)(), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#)().

#### 25.33.4.9 stdair::AirlineCodeList\_T AIRTSP::OnDPeriodStruct::\_airlineCodeList

Definition at line 50 of file [OnDPeriodStruct.hpp](#).

Referenced by [getFirstAirlineCode\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#)(), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#)().

#### 25.33.4.10 stdair::ClassCodeList\_T AIRTSP::OnDPeriodStruct::\_classCodeList

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

Referenced by [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#)().

#### 25.33.4.11 stdair::Date\_T AIRTSP::OnDPeriodStruct::\_dateRangeStart

Staging Date.

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

Referenced by [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#)().

#### 25.33.4.12 stdair::Date\_T AIRTSP::OnDPeriodStruct::\_dateRangeEnd

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

Referenced by [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#)().

#### 25.33.4.13 unsigned int AIRTSP::OnDPeriodStruct::\_itYear

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

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.33.4.14 unsigned int AIRTSP::OnDPeriodStruct::\_itMonth

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

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.33.4.15 unsigned int AIRTSP::OnDPeriodStruct::\_itDay

Definition at line 58 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [getDate\(\)](#).

#### 25.33.4.16 long AIRTSP::OnDPeriodStruct::\_itHours

Staging Time.

Definition at line 61 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [getTime\(\)](#).

#### 25.33.4.17 long AIRTSP::OnDPeriodStruct::\_itMinutes

Definition at line 62 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), and [getTime\(\)](#).

#### 25.33.4.18 long AIRTSP::OnDPeriodStruct::\_itSeconds

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

Referenced by [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#), [getTime\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#) and [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#).

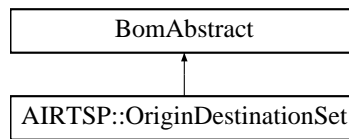
The documentation for this struct was generated from the following files:

- [airtsp/bom/OnDPeriodStruct.hpp](#)
- [airtsp/bom/OnDPeriodStruct.cpp](#)

## 25.34 AIRTSP::OriginDestinationSet Class Reference

Class representing a simple sub-network.

`#include <airtsp/bom/OriginDestinationSet.hpp>`  
 Inheritance diagram for AIRTSP::OriginDestinationSet::



## Public Types

- typedef [OriginDestinationSetKey](#) [Key\\_T](#)

## Public Member Functions

- const [Key\\_T](#) & [getKey](#) () const
- const stdair::AirportCode\_T & [getDestination](#) () const
- stdair::BomAbstract \*const [getParent](#) () const
- const stdair::HolderMap\_T & [getHolderMap](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

## Protected Member Functions

- [OriginDestinationSet](#) (const [Key\\_T](#) &)
- [~OriginDestinationSet](#) ()

## Protected Attributes

- [Key\\_T \\_key](#)
- stdair::BomAbstract \* [\\_parent](#)
- stdair::HolderMap\_T [\\_holderMap](#)

## Friends

- class stdair::FacBom
- class stdair::FacBomManager
- class boost::serialization::access

### 25.34.1 Detailed Description

Class representing a simple sub-network. That simple sub-network is made of a set of segments ([SegmentPathPeriod](#) objects), from the origin airport specified within [ReachableUniverse](#) (parent object) to the destination specified in the [OriginDestinationSetKey](#) object.

Each segment (composing that [OriginDestinationSet](#) object) corresponds to an actual travel solution from the origin to the destination, that is, a path that a traveller can take with actual scheduled flights.

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

## 25.34.2 Member Typedef Documentation

### 25.34.2.1 typedef OriginDestinationSetKey AIRTSP::OriginDestinationSet::Key\_T

Definition allowing to retrieve the associated BOM key type.

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

## 25.34.3 Constructor & Destructor Documentation

### 25.34.3.1 AIRTSP::OriginDestinationSet::OriginDestinationSet (const Key\_T & iKey) [protected]

Main constructor.

Definition at line 31 of file [OriginDestinationSet.cpp](#).

### 25.34.3.2 AIRTSP::OriginDestinationSet::~~OriginDestinationSet () [protected]

Destructor.

Definition at line 36 of file [OriginDestinationSet.cpp](#).

## 25.34.4 Member Function Documentation

### 25.34.4.1 const Key\_T& AIRTSP::OriginDestinationSet::getKey () const [inline]

Get the primary key (destination airport).

Definition at line 65 of file [OriginDestinationSet.hpp](#).

References [\\_key](#).

### 25.34.4.2 const stdair::AirportCode\_T& AIRTSP::OriginDestinationSet::getDestination () const [inline]

Get the destination airport (i.e., the primary key).

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

References [\\_key](#), and [AIRTSP::OriginDestinationSetKey::getOffPoint\(\)](#).

### 25.34.4.3 stdair::BomAbstract\* const AIRTSP::OriginDestinationSet::getParent () const [inline]

Get the parent (i.e., [ReachableUniverse](#)) object.

Definition at line 79 of file [OriginDestinationSet.hpp](#).

References [\\_parent](#).

**25.34.4.4** `const stdair::HolderMap_T& AIRTSP::OriginDestinationSet::getHolderMap () const [inline]`

Get the map of children holders ([SegmentPathPeriod](#) objects).

Definition at line 86 of file [OriginDestinationSet.hpp](#).

References [\\_holderMap](#).

**25.34.4.5** `void AIRTSP::OriginDestinationSet::toStream (std::ostream & ioOut) const [inline]`

Dump a Business Object into an output stream.

**Parameters:**

*ostream&* the output stream.

Definition at line 98 of file [OriginDestinationSet.hpp](#).

References [toString\(\)](#).

**25.34.4.6** `void AIRTSP::OriginDestinationSet::fromStream (std::istream & ioIn) [inline]`

Read a Business Object from an input stream.

**Parameters:**

*istream&* the input stream.

Definition at line 107 of file [OriginDestinationSet.hpp](#).

**25.34.4.7** `std::string AIRTSP::OriginDestinationSet::toString () const`

Get the serialised version of the Business Object.

Definition at line 40 of file [OriginDestinationSet.cpp](#).

References [\\_key](#), and [AIRTSP::OriginDestinationSetKey::toString\(\)](#).

Referenced by [toStream\(\)](#).

**25.34.4.8** `const std::string AIRTSP::OriginDestinationSet::describeKey () const [inline]`

Get a string describing the key.

Definition at line 118 of file [OriginDestinationSet.hpp](#).

References [\\_key](#), and [AIRTSP::OriginDestinationSetKey::toString\(\)](#).

**25.34.4.9** `template<class Archive > void AIRTSP::OriginDestinationSet::serialize (Archive & ar, const unsigned int iFileVersion) [inline]`

Serialisation.



Definition at line 62 of file [OriginDestinationSet.cpp](#).

References [\\_key](#).

### 25.34.5 Friends And Related Function Documentation

#### 25.34.5.1 friend class stdair::FacBom [friend]

Friend classes.

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

#### 25.34.5.2 friend class stdair::FacBomManager [friend]

Definition at line 49 of file [OriginDestinationSet.hpp](#).

#### 25.34.5.3 friend class boost::serialization::access [friend]

Definition at line 50 of file [OriginDestinationSet.hpp](#).

### 25.34.6 Member Data Documentation

#### 25.34.6.1 Key\_T AIRTSP::OriginDestinationSet::\_key [protected]

Primary key (destination airport code).

Definition at line 168 of file [OriginDestinationSet.hpp](#).

Referenced by [describeKey\(\)](#), [getDestination\(\)](#), [getKey\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

#### 25.34.6.2 stdair::BomAbstract\* AIRTSP::OriginDestinationSet::\_parent [protected]

Pointer on the parent ([ReachableUniverse](#)) object.

Definition at line 173 of file [OriginDestinationSet.hpp](#).

Referenced by [getParent\(\)](#).

#### 25.34.6.3 stdair::HolderMap\_T AIRTSP::OriginDestinationSet::\_holderMap [protected]

Map holding the children ([SegmentPathPeriod](#) objects).

Definition at line 178 of file [OriginDestinationSet.hpp](#).

Referenced by [getHolderMap\(\)](#).

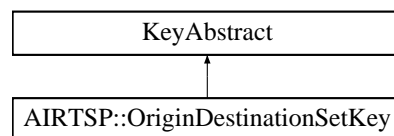
The documentation for this class was generated from the following files:

- [airtsp/bom/OriginDestinationSet.hpp](#)
- [airtsp/bom/OriginDestinationSet.cpp](#)

## 25.35 AIRTSP::OriginDestinationSetKey Struct Reference

Structure representing the key of a sub-network.

#include <airtsp/bom/OriginDestinationSetKey.hpp> Inheritance diagram for AIRTSP::OriginDestinationSetKey::



### Public Member Functions

- [OriginDestinationSetKey](#) (const stdair::AirportCode\_T &iDestination)
- [OriginDestinationSetKey](#) (const [OriginDestinationSetKey](#) &)
- [~OriginDestinationSetKey](#) ()
- const stdair::AirportCode\_T & [getOffPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

### Friends

- class [boost::serialization::access](#)

### 25.35.1 Detailed Description

Structure representing the key of a sub-network. As the origin airport code is already part of the [ReachableUniverse](#) (parent) class, that key is only made of the destination airport code.

Definition at line 30 of file [OriginDestinationSetKey.hpp](#).

### 25.35.2 Constructor & Destructor Documentation

#### 25.35.2.1 AIRTSP::OriginDestinationSetKey::OriginDestinationSetKey (const stdair::AirportCode\_T &iDestination)

Constructor.

Definition at line 26 of file [OriginDestinationSetKey.cpp](#).

#### 25.35.2.2 AIRTSP::OriginDestinationSetKey::OriginDestinationSetKey (const OriginDestinationSetKey &iKey)

Copy constructor.

Definition at line 32 of file [OriginDestinationSetKey.cpp](#).

### 25.35.2.3 AIRTSP::OriginDestinationSetKey::~~OriginDestinationSetKey ()

Destructor.

Definition at line 37 of file [OriginDestinationSetKey.cpp](#).

### 25.35.3 Member Function Documentation

#### 25.35.3.1 const std::air::AirportCode\_T& AIRTSP::OriginDestinationSetKey::getOffPoint () const [inline]

Get the destination airport.

Definition at line 62 of file [OriginDestinationSetKey.hpp](#).

Referenced by [AIRTSP::OriginDestinationSet::getDestination\(\)](#).

#### 25.35.3.2 void AIRTSP::OriginDestinationSetKey::toStream (std::ostream & ioOut) const

Dump a Business Object Key into an output stream.

##### Parameters:

*ostream&* the output stream.

Definition at line 41 of file [OriginDestinationSetKey.cpp](#).

References [toString\(\)](#).

#### 25.35.3.3 void AIRTSP::OriginDestinationSetKey::fromStream (std::istream & ioIn)

Read a Business Object Key from an input stream.

##### Parameters:

*istream&* the input stream.

Definition at line 46 of file [OriginDestinationSetKey.cpp](#).

#### 25.35.3.4 const std::string AIRTSP::OriginDestinationSetKey::toString () const

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Definition at line 50 of file [OriginDestinationSetKey.cpp](#).

Referenced by [AIRTSP::OriginDestinationSet::describeKey\(\)](#), [toStream\(\)](#), and [AIRTSP::OriginDestinationSet::toString\(\)](#).

**25.35.3.5** `template<class Archive > void AIRTSP::OriginDestinationSetKey::serialize (Archive & ar, const unsigned int iFileVersion) [inline]`

Serialisation.

Definition at line 72 of file [OriginDestinationSetKey.cpp](#).

## 25.35.4 Friends And Related Function Documentation

**25.35.4.1** `friend class boost::serialization::access [friend]`

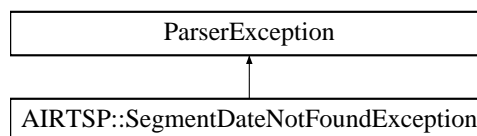
Definition at line 31 of file [OriginDestinationSetKey.hpp](#).

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

- [airtsp/bom/OriginDestinationSetKey.hpp](#)
- [airtsp/bom/OriginDestinationSetKey.cpp](#)

## 25.36 ParserException Class Reference

Inheritance diagram for ParserException::

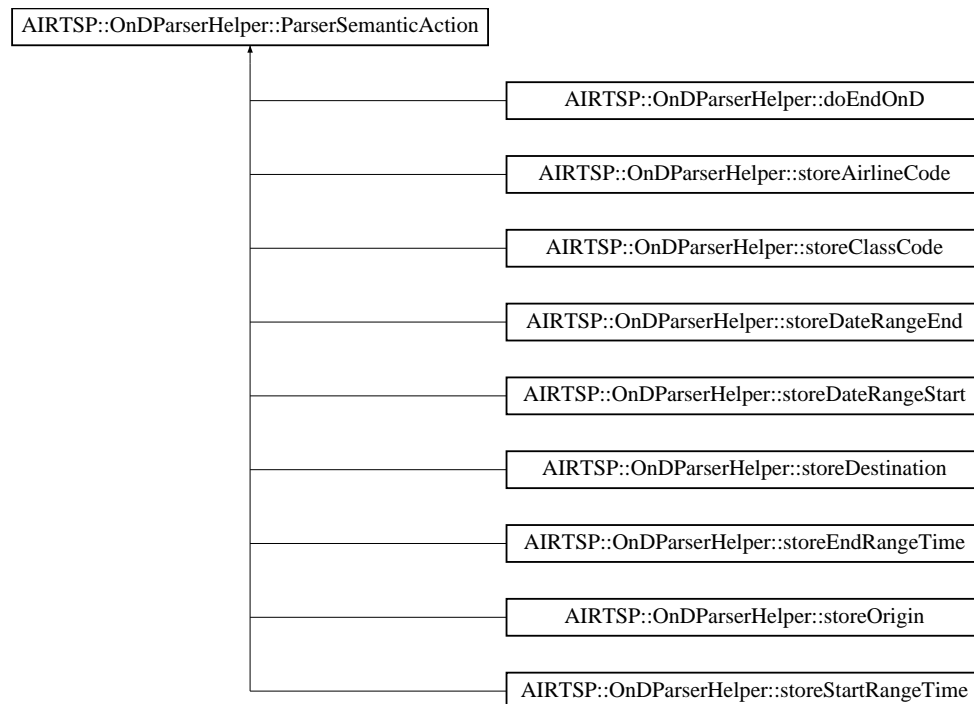


The documentation for this class was generated from the following file:

- [airtsp/AIRTSP\\_Types.hpp](#)

## 25.37 AIRTSP::OnDParserHelper::ParserSemanticAction Struct Reference

`#include <airtsp/command/OnDParserHelper.hpp>`  
 Inheritance diagram for AIRTSP::OnDParserHelper::ParserSemanticAction::



## Public Member Functions

- [ParserSemanticAction](#) ([OnDPeriodStruct](#) &)

## Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

### 25.37.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

Definition at line 34 of file [OnDParserHelper.hpp](#).

### 25.37.2 Constructor & Destructor Documentation

#### 25.37.2.1 AIRTSP::OnDParserHelper::ParserSemanticAction::ParserSemanticAction (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 25 of file [OnDParserHelper.cpp](#).

### 25.37.3 Member Data Documentation

#### 25.37.3.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

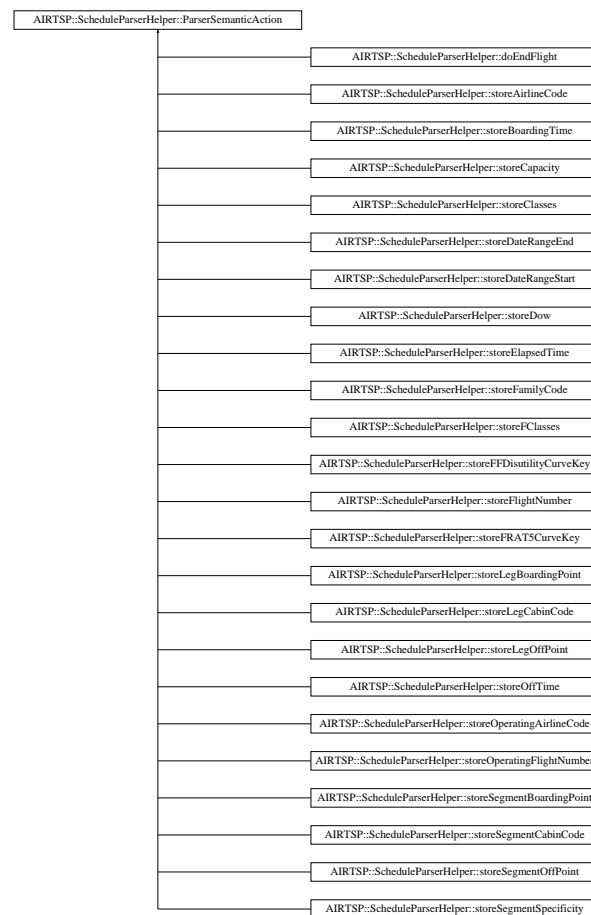
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.38 AIRTSP::ScheduleParserHelper::ParserSemanticAction Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>`Inheritance diagram for AIRTSP::ScheduleParserHelper::ParserSemanticAction:



### Public Member Functions

- [ParserSemanticAction](#) ([FlightPeriodStruct](#) &)

## Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.38.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

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

### 25.38.2 Constructor & Destructor Documentation

#### 25.38.2.1 AIRTSP::ScheduleParserHelper::ParserSemanticAction::ParserSemanticAction (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 26 of file [ScheduleParserHelper.cpp](#).

### 25.38.3 Member Data Documentation

#### 25.38.3.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_- flightPeriod

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#) and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.39 airtsp::Passenger\_T Struct Reference

```
#include <airtsp/batches/BookingRequestParser.hpp>
```

## Public Types

- enum [PassengerType\\_T](#) { [ADULT](#) = 0, [CHILD](#), [PET](#), [LAST\\_VALUE](#) }

## Public Member Functions

- [Passenger\\_T](#) ()
- void [display](#) () const

## Public Attributes

- [PassengerType\\_T](#) \_type
- unsigned short [\\_number](#)

## Static Public Attributes

- static const std::string [\\_labels](#) [[LAST\\_VALUE](#)]

### 25.39.1 Detailed Description

Passenger.

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

### 25.39.2 Member Enumeration Documentation

#### 25.39.2.1 enum airtsp::Passenger\_T::PassengerType\_T

##### Enumerator:

*ADULT*

*CHILD*

*PET*

*LAST\_VALUE*

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

### 25.39.3 Constructor & Destructor Documentation

#### 25.39.3.1 airtsp::Passenger\_T::Passenger\_T () [inline]

Constructor.

Definition at line 78 of file [BookingRequestParser.hpp](#).



### 25.39.4 Member Function Documentation

#### 25.39.4.1 void airtsp::Passenger\_T::display () const [inline]

Definition at line 80 of file [BookingRequestParser.hpp](#).

References [\\_labels](#), [\\_number](#), and [\\_type](#).

Referenced by [airtsp::SearchString\\_T::display\(\)](#).

### 25.39.5 Member Data Documentation

#### 25.39.5.1 const std::string airtsp::Passenger\_T::\_labels [static]

Initial value:

```
{ "Adult", "Child", "Pet" }
```

Passenger type labels.

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

Referenced by [display\(\)](#).

#### 25.39.5.2 PassengerType\_T airtsp::Passenger\_T::\_type

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

Referenced by [display\(\)](#), [airtsp::store\\_pet\\_passenger\\_type::operator\(\)](#), [airtsp::store\\_child\\_passenger\\_type::operator\(\)](#), and [airtsp::store\\_adult\\_passenger\\_type::operator\(\)](#).

#### 25.39.5.3 unsigned short airtsp::Passenger\_T::\_number

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

Referenced by [display\(\)](#), and [airtsp::store\\_passenger\\_number::operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.hpp](#)

## 25.40 airtsp::Place\_T Struct Reference

```
#include <airtsp/batches/BookingRequestParser.hpp>
```

### Public Member Functions

- [Place\\_T \(\)](#)
- void [display \(\)](#) const

## Public Attributes

- [std::string \\_name](#)
- [std::string \\_code](#)

### 25.40.1 Detailed Description

Place.

Definition at line 11 of file [BookingRequestParser.hpp](#).

### 25.40.2 Constructor & Destructor Documentation

#### 25.40.2.1 airtsp::Place\_T::Place\_T () [\[inline\]](#)

Constructor.

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

### 25.40.3 Member Function Documentation

#### 25.40.3.1 void airtsp::Place\_T::display () const [\[inline\]](#)

Definition at line 18 of file [BookingRequestParser.hpp](#).

References [\\_code](#), and [\\_name](#).

Referenced by [airtsp::SearchString\\_T::display\(\)](#).

### 25.40.4 Member Data Documentation

#### 25.40.4.1 std::string airtsp::Place\_T::\_name

Definition at line 13 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), and [airtsp::store\\_place\\_element::operator\(\)](#).

#### 25.40.4.2 std::string airtsp::Place\_T::\_code

Definition at line 14 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.hpp](#)

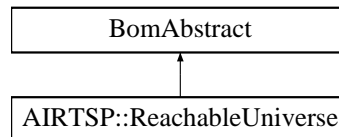
## 25.41 AIRTSP::ReachableUniverse Class Reference

Class representing the root of the schedule-related BOM tree.

`#include <airtsp/bom/ReachableUniverse.hpp>` Inheritance  
AIRTSP::ReachableUniverse::

diagram

for



### Public Types

- typedef [ReachableUniverseKey](#) [Key\\_T](#)

### Public Member Functions

- const [Key\\_T](#) & [getKey](#) () const
- const stdair::AirportCode\_T & [getOrigin](#) () const
- stdair::BomAbstract \*const [getParent](#) () const
- const stdair::HolderMap\_T & [getHolderMap](#) () const
- const [SegmentPathPeriodListList\\_T](#) & [getSegmentPathPeriodListList](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

### Protected Member Functions

- [ReachableUniverse](#) (const [Key\\_T](#) &)
- [~ReachableUniverse](#) ()

### Protected Attributes

- [Key\\_T \\_key](#)
- stdair::BomAbstract \* [\\_parent](#)
- stdair::HolderMap\_T [\\_holderMap](#)
- [SegmentPathPeriodListList\\_T \\_segmentPathPeriodListList](#)

### Friends

- class [stdair::FacBom](#)
- class [stdair::FacBomManager](#)
- class [SegmentPathGenerator](#)
- class [boost::serialization::access](#)

### 25.41.1 Detailed Description

Class representing the root of the schedule-related BOM tree. It is the pending, in the schedule universe, of the `stdair::Inventory` class. It corresponds to all the destinations, which can be reached from a given geographical point. That latter is an airport for now, and its key (airport code) is specified by the [ReachableUniverseKey](#) object.

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

### 25.41.2 Member Typedef Documentation

#### 25.41.2.1 `typedef ReachableUniverseKey AIRTSP::ReachableUniverse::Key_T`

Definition allowing to retrieve the associated BOM key type.

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

### 25.41.3 Constructor & Destructor Documentation

#### 25.41.3.1 `AIRTSP::ReachableUniverse::ReachableUniverse (const Key_T & iKey) [protected]`

Main constructor.

Definition at line 32 of file [ReachableUniverse.cpp](#).

#### 25.41.3.2 `AIRTSP::ReachableUniverse::~~ReachableUniverse () [protected]`

Destructor.

Definition at line 37 of file [ReachableUniverse.cpp](#).

### 25.41.4 Member Function Documentation

#### 25.41.4.1 `const Key_T& AIRTSP::ReachableUniverse::getKey () const [inline]`

Get the universe key (airport code representing the departure point of the "reachable universe").

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

References [\\_key](#).

#### 25.41.4.2 `const stdair::AirportCode_T& AIRTSP::ReachableUniverse::getOrigin () const [inline]`

Get the (origin) airport (i.e., the primary key).

Definition at line 70 of file [ReachableUniverse.hpp](#).

References [\\_key](#), and [AIRTSP::ReachableUniverseKey::getBoardingPoint\(\)](#).

**25.41.4.3 stdair::BomAbstract\* const AIRTSP::ReachableUniverse::getParent () const [inline]**

Get the parent (i.e., the BomRoot) object.

Definition at line 77 of file [ReachableUniverse.hpp](#).

References [\\_parent](#).

**25.41.4.4 const stdair::HolderMap\_T& AIRTSP::ReachableUniverse::getHolderMap () const [inline]**

Get the map of children holders ([OriginDestinationSet](#) objects).

Definition at line 84 of file [ReachableUniverse.hpp](#).

References [\\_holderMap](#).

**25.41.4.5 const SegmentPathPeriodListList\_T& AIRTSP::ReachableUniverse::getSegmentPathPeriodListList () const [inline]**

Get the vector of SegmentPathPeriodLightList objects.

Definition at line 91 of file [ReachableUniverse.hpp](#).

References [\\_segmentPathPeriodListList](#).

**25.41.4.6 void AIRTSP::ReachableUniverse::toStream (std::ostream & ioOut) const [inline]**

Dump a Business Object into an output stream.

**Parameters:**

*ostream&* the output stream.

Definition at line 103 of file [ReachableUniverse.hpp](#).

References [toString\(\)](#).

**25.41.4.7 void AIRTSP::ReachableUniverse::fromStream (std::istream & ioIn) [inline]**

Read a Business Object from an input stream.

**Parameters:**

*istream&* the input stream.

Definition at line 112 of file [ReachableUniverse.hpp](#).

**25.41.4.8 std::string AIRTSP::ReachableUniverse::toString () const**

Get the serialised version of the Business Object.

Definition at line 41 of file [ReachableUniverse.cpp](#).

References [\\_key](#), and [AIRTSP::ReachableUniverseKey::toString\(\)](#).

Referenced by [AIRTSP::BomDisplay::csvDisplay\(\)](#), and [toStream\(\)](#).

#### 25.41.4.9 `const std::string AIRTSP::ReachableUniverse::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 123 of file [ReachableUniverse.hpp](#).

References [\\_key](#), and [AIRTSP::ReachableUniverseKey::toString\(\)](#).

#### 25.41.4.10 `template<class Archive > void AIRTSP::ReachableUniverse::serialize (Archive & ar, const unsigned int iFileVersion) [inline]`

Serialisation.

Definition at line 63 of file [ReachableUniverse.cpp](#).

References [\\_key](#).

### 25.41.5 Friends And Related Function Documentation

#### 25.41.5.1 `friend class stdair::FacBom` `[friend]`

Friend classes.

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

#### 25.41.5.2 `friend class stdair::FacBomManager` `[friend]`

Definition at line 46 of file [ReachableUniverse.hpp](#).

#### 25.41.5.3 `friend class SegmentPathGenerator` `[friend]`

Definition at line 47 of file [ReachableUniverse.hpp](#).

#### 25.41.5.4 `friend class boost::serialization::access` `[friend]`

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

### 25.41.6 Member Data Documentation

#### 25.41.6.1 `Key_T AIRTSP::ReachableUniverse::_key` `[protected]`

Primary key (origin airport code).

Definition at line 174 of file [ReachableUniverse.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [getOrigin\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

#### 25.41.6.2 stdair::BomAbstract\* AIRTSP::ReachableUniverse::\_parent [protected]

Pointer on the parent (BomRoot) object.

Definition at line 179 of file [ReachableUniverse.hpp](#).

Referenced by [getParent\(\)](#).

#### 25.41.6.3 stdair::HolderMap\_T AIRTSP::ReachableUniverse::\_holderMap [protected]

Map holding the children ([OriginDestinationSet](#) objects).

Definition at line 184 of file [ReachableUniverse.hpp](#).

Referenced by [getHolderMap\(\)](#).

#### 25.41.6.4 SegmentPathPeriodListList\_T AIRTSP::ReachableUniverse::\_segmentPathPeriodListList [protected]

The list (actually, a vector) of lists of SegmentPathPeriods, used solely for the construction of the main list of SegmentPathPeriods within the ReachableUniverseStructure.

Definition at line 191 of file [ReachableUniverse.hpp](#).

Referenced by [getSegmentPathPeriodListList\(\)](#).

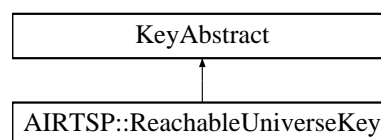
The documentation for this class was generated from the following files:

- [airtsp/bom/ReachableUniverse.hpp](#)
- [airtsp/bom/ReachableUniverse.cpp](#)

## 25.42 AIRTSP::ReachableUniverseKey Struct Reference

Structure representing the key of the schedule-related BOM tree root.

#include <[airtsp/bom/ReachableUniverseKey.hpp](#)> Inheritance diagram for AIRTSP::ReachableUniverseKey:



### Public Member Functions

- [ReachableUniverseKey](#) (const stdair::AirportCode\_T &iOrigin)
- [ReachableUniverseKey](#) (const [ReachableUniverseKey](#) &)
- [~ReachableUniverseKey](#) ()
- const stdair::AirportCode\_T & [getBoardingPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const

- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

## Friends

- class [boost::serialization::access](#)

### 25.42.1 Detailed Description

Structure representing the key of the schedule-related BOM tree root. The [ReachableUniverse](#) is the pending, in the schedule universe, of the `stdair::Inventory` class. It corresponds to all the destinations which can be reached from a given geographical point. That latter is an airport for now, and the present structure specifies its key (i.e., airport code).

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

### 25.42.2 Constructor & Destructor Documentation

#### 25.42.2.1 AIRTSP::ReachableUniverseKey::ReachableUniverseKey (const stdair::AirportCode\_T & iOrigin)

Constructor.

Definition at line 32 of file [ReachableUniverseKey.cpp](#).

#### 25.42.2.2 AIRTSP::ReachableUniverseKey::ReachableUniverseKey (const ReachableUniverseKey & iKey)

Copy constructor.

Definition at line 26 of file [ReachableUniverseKey.cpp](#).

#### 25.42.2.3 AIRTSP::ReachableUniverseKey::~~ReachableUniverseKey ()

Destructor.

Definition at line 37 of file [ReachableUniverseKey.cpp](#).

### 25.42.3 Member Function Documentation

#### 25.42.3.1 const stdair::AirportCode\_T& AIRTSP::ReachableUniverseKey::getBoardingPoint () const [inline]

Get the origin airport (from which the remaining universe may be reached).

Definition at line 66 of file [ReachableUniverseKey.hpp](#).

Referenced by [AIRTSP::ReachableUniverse::getOrigin\(\)](#).



### 25.42.3.2 void AIRTSP::ReachableUniverseKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

#### Parameters:

*ostream&* the output stream.

Definition at line 41 of file [ReachableUniverseKey.cpp](#).

References [toString\(\)](#).

### 25.42.3.3 void AIRTSP::ReachableUniverseKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

#### Parameters:

*istream&* the input stream.

Definition at line 46 of file [ReachableUniverseKey.cpp](#).

### 25.42.3.4 const std::string AIRTSP::ReachableUniverseKey::toString () const

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Definition at line 50 of file [ReachableUniverseKey.cpp](#).

Referenced by [AIRTSP::ReachableUniverse::describeKey\(\)](#), [toStream\(\)](#), and [AIRTSP::ReachableUniverse::toString\(\)](#).

### 25.42.3.5 template<class Archive > void AIRTSP::ReachableUniverseKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 72 of file [ReachableUniverseKey.cpp](#).

## 25.42.4 Friends And Related Function Documentation

### 25.42.4.1 friend class boost::serialization::access [friend]

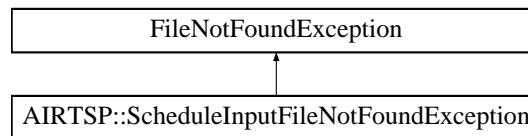
Definition at line 34 of file [ReachableUniverseKey.hpp](#).

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

- [airtsp/bom/ReachableUniverseKey.hpp](#)
- [airtsp/bom/ReachableUniverseKey.cpp](#)

## 25.43 AIRTSP::ScheduleInputFileNotFoundException Class Reference

#include <airtsp/AIRTSP\_Types.hpp> Inheritance diagram for AIRTSP::ScheduleInputFileNotFoundException::



### Public Member Functions

- [ScheduleInputFileNotFoundException](#) (const std::string &iWhat)

#### 25.43.1 Detailed Description

The schedule input file cannot be retrieved.

Definition at line 47 of file [AIRTSP\\_Types.hpp](#).

#### 25.43.2 Constructor & Destructor Documentation

##### 25.43.2.1 AIRTSP::ScheduleInputFileNotFoundException::ScheduleInputFileNotFoundException (const std::string & iWhat) [inline]

Constructor.

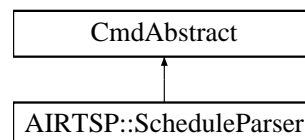
Definition at line 53 of file [AIRTSP\\_Types.hpp](#).

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

- [airtsp/AIRTSP\\_Types.hpp](#)

## 25.44 AIRTSP::ScheduleParser Class Reference

#include <airtsp/command/ScheduleParser.hpp> Inheritance diagram for AIRTSP::ScheduleParser::



### Static Public Member Functions

- static void [generateInventories](#) (const stdair::ScheduleFilePath &, stdair::BomRoot &)

### 25.44.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 22 of file [ScheduleParser.hpp](#).

### 25.44.2 Member Function Documentation

#### 25.44.2.1 void AIRTSP::ScheduleParser::generateInventories (const stdair::ScheduleFilePath & iScheduleFilename, stdair::BomRoot & ioBomRoot) [static]

Parse the CSV file describing the airline schedules for the simulator, and generates the inventories accordingly.

#### Parameters:

- const** stdair::ScheduleFilePath& The file-name of the CSV-formatted schedule input file.
- stdair::BomRoot&** Root of the BOM tree.

Definition at line 20 of file [ScheduleParser.cpp](#).

References [AIRTSP::FlightPeriodFileParser::generateInventories\(\)](#).

Referenced by [AIRTSP::AIRTSP\\_Service::parseAndLoad\(\)](#).

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

- [airtsp/command/ScheduleParser.hpp](#)
- [airtsp/command/ScheduleParser.cpp](#)

## 25.45 airtsp::SearchString\_T Struct Reference

```
#include <airtsp/batches/BookingRequestParser.hpp>
```

### Public Member Functions

- [SearchString\\_T\(\)](#)
- void [display\(\)](#) const

### Public Attributes

- [PlaceList\\_T \\_placeList](#)
- [DateList\\_T \\_dateList](#)
- [AirlineList\\_T \\_airlineList](#)
- [PassengerList\\_T \\_passengerList](#)
- [Place\\_T \\_tmpPlace](#)
- [Date\\_T \\_tmpDate](#)
- [Airline\\_T \\_tmpAirline](#)
- [Passenger\\_T \\_tmpPassenger](#)

### 25.45.1 Detailed Description

Search string.

Definition at line 94 of file [BookingRequestParser.hpp](#).

### 25.45.2 Constructor & Destructor Documentation

#### 25.45.2.1 airtsp::SearchString\_T::SearchString\_T () [inline]

Constructor.

Definition at line 102 of file [BookingRequestParser.hpp](#).

### 25.45.3 Member Function Documentation

#### 25.45.3.1 void airtsp::SearchString\_T::display () const [inline]

Definition at line 105 of file [BookingRequestParser.hpp](#).

References [\\_airlineList](#), [\\_dateList](#), [\\_passengerList](#), [\\_placeList](#), [\\_tmpPlace](#), [airtsp::Passenger\\_T::display\(\)](#), [airtsp::Airline\\_T::display\(\)](#), [airtsp::Date\\_T::display\(\)](#), and [airtsp::Place\\_T::display\(\)](#).

### 25.45.4 Member Data Documentation

#### 25.45.4.1 PlaceList\_T airtsp::SearchString\_T::\_placeList

Definition at line 96 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#).

#### 25.45.4.2 DateList\_T airtsp::SearchString\_T::\_dateList

Definition at line 97 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), and [airtsp::store\\_date::operator\(\)](#).

#### 25.45.4.3 AirlineList\_T airtsp::SearchString\_T::\_airlineList

Definition at line 98 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), [airtsp::store\\_airline\\_name::operator\(\)](#), and [airtsp::store\\_airline\\_code::operator\(\)](#).

#### 25.45.4.4 PassengerList\_T airtsp::SearchString\_T::\_passengerList

Definition at line 99 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), [airtsp::store\\_pet\\_passenger\\_type::operator\(\)](#), [airtsp::store\\_child\\_passenger\\_type::operator\(\)](#), and [airtsp::store\\_adult\\_passenger\\_type::operator\(\)](#).

#### 25.45.4.5 Place\_T airtsp::SearchString\_T::\_tmpPlace

Definition at line 137 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), and [airtsp::store\\_place\\_element::operator\(\)](#).

#### 25.45.4.6 Date\_T airtsp::SearchString\_T::\_tmpDate

Definition at line 138 of file [BookingRequestParser.hpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#), and [airtsp::store\\_date::operator\(\)](#).

#### 25.45.4.7 Airline\_T airtsp::SearchString\_T::\_tmpAirline

Definition at line 139 of file [BookingRequestParser.hpp](#).

Referenced by [airtsp::store\\_airline\\_name::operator\(\)](#), [airtsp::store\\_airline\\_code::operator\(\)](#), and [airtsp::store\\_airline\\_sign::operator\(\)](#).

#### 25.45.4.8 Passenger\_T airtsp::SearchString\_T::\_tmpPassenger

Definition at line 140 of file [BookingRequestParser.hpp](#).

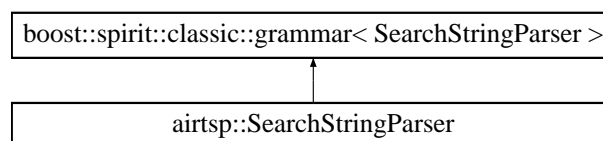
Referenced by [airtsp::store\\_pet\\_passenger\\_type::operator\(\)](#), [airtsp::store\\_child\\_passenger\\_type::operator\(\)](#), [airtsp::store\\_adult\\_passenger\\_type::operator\(\)](#), and [airtsp::store\\_passenger\\_number::operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.hpp](#)

## 25.46 airtsp::SearchStringParser Struct Reference

Inheritance diagram for airtsp::SearchStringParser:



**Classes**

- struct [definition](#)

**Public Member Functions**

- [SearchStringParser](#) ([SearchString\\_T](#) &[ioSearchString](#))

**Public Attributes**

- [SearchString\\_T](#) & [\\_searchString](#)

**25.46.1 Detailed Description**

Grammar for the search string parser.

Definition at line 251 of file [BookingRequestParser.cpp](#).

**25.46.2 Constructor & Destructor Documentation****25.46.2.1 airtsp::SearchStringParser::SearchStringParser ([SearchString\\_T](#) & *ioSearchString*)  
[[inline](#)]**

Definition at line 254 of file [BookingRequestParser.cpp](#).

**25.46.3 Member Data Documentation****25.46.3.1 [SearchString\\_T](#) & airtsp::SearchStringParser::\_searchString**

Definition at line 369 of file [BookingRequestParser.cpp](#).

Referenced by [airtsp::SearchStringParser::definition< ScannerT >::definition\(\)](#).

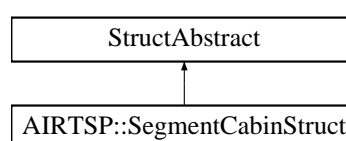
The documentation for this struct was generated from the following file:

- airtsp/batches/[BookingRequestParser.cpp](#)

**25.47 AIRTSP::SegmentCabinStruct Struct Reference**

```
#include <airtsp/bom/SegmentCabinStruct.hpp>
Inheritance
AIRTSP::SegmentCabinStruct::
```

diagram for



## Public Member Functions

- void [fill](#) (stdair::SegmentCabin &) const
- const std::string [describe](#) () const

## Public Attributes

- stdair::CabinCode\_T [\\_cabinCode](#)
- stdair::ClassList\_String\_T [\\_classes](#)
- stdair::FamilyCode\_T [\\_itFamilyCode](#)
- stdair::CurveKey\_T [\\_itFRAT5CurveKey](#)
- stdair::CurveKey\_T [\\_itFFDisutilityCurveKey](#)
- [FareFamilyStructList\\_T](#) [\\_fareFamilies](#)

### 25.47.1 Detailed Description

Utility Structure for the parsing of SegmentCabin details.

Definition at line 24 of file [SegmentCabinStruct.hpp](#).

### 25.47.2 Member Function Documentation

#### 25.47.2.1 void AIRTSP::SegmentCabinStruct::fill (stdair::SegmentCabin & *ioSegmentCabin*) const

Fill the SegmentCabin objects with the attributes of the [SegmentCabinStruct](#).

Definition at line 22 of file [SegmentCabinStruct.cpp](#).

#### 25.47.2.2 const std::string AIRTSP::SegmentCabinStruct::describe () const

Give a description of the structure (for display purposes).

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

References [\\_cabinCode](#), and [\\_classes](#).

Referenced by [AIRTSP::SegmentStruct::describe\(\)](#).

### 25.47.3 Member Data Documentation

#### 25.47.3.1 stdair::CabinCode\_T AIRTSP::SegmentCabinStruct::\_cabinCode

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

Referenced by [AIRTSP::FlightPeriodStruct::addFareFamily\(\)](#), [describe\(\)](#), [AIRTSP::SegmentPeriodHelper::fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#).

**25.47.3.2 stdair::ClassList\_String\_T AIRTSP::SegmentCabinStruct::\_classes**

Definition at line 27 of file [SegmentCabinStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRTSP::SegmentPeriodHelper::fill\(\)](#), and [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#).

**25.47.3.3 stdair::FamilyCode\_T AIRTSP::SegmentCabinStruct::\_itFamilyCode**

Definition at line 28 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#).

**25.47.3.4 stdair::CurveKey\_T AIRTSP::SegmentCabinStruct::\_itFRAT5CurveKey**

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

Referenced by [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#).

**25.47.3.5 stdair::CurveKey\_T AIRTSP::SegmentCabinStruct::\_itFFDisutilityCurveKey**

Definition at line 30 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#).

**25.47.3.6 FareFamilyStructList\_T AIRTSP::SegmentCabinStruct::\_fareFamilies**

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

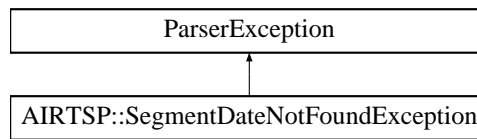
The documentation for this struct was generated from the following files:

- [airtsp/bom/SegmentCabinStruct.hpp](#)
- [airtsp/bom/SegmentCabinStruct.cpp](#)

**25.48 AIRTSP::SegmentDateNotFoundException Class Reference**

`#include <airtsp/AIRTSP_Types.hpp>` Inheritance diagram for AIRTSP::SegmentDateNotFoundException::





### Public Member Functions

- [SegmentDateNotFoundException](#) (const std::string &iWhat)

#### 25.48.1 Detailed Description

Specific exception when some BOM objects can not be found within the schedule.

Definition at line 23 of file [AIRTSP\\_Types.hpp](#).

#### 25.48.2 Constructor & Destructor Documentation

##### 25.48.2.1 AIRTSP::SegmentDateNotFoundException::SegmentDateNotFoundException (const std::string &iWhat) [inline]

Constructor.

Definition at line 28 of file [AIRTSP\\_Types.hpp](#).

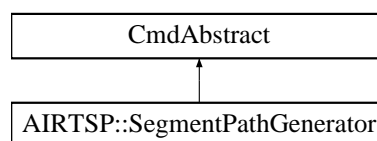
The documentation for this class was generated from the following file:

- [airtsp/AIRTSP\\_Types.hpp](#)

## 25.49 AIRTSP::SegmentPathGenerator Class Reference

Class handling the generation / instantiation of the network BOM.

`#include <airtsp/command/SegmentPathGenerator.hpp>`  
 Inheritance diagram for AIRTSP::SegmentPathGenerator::



### Static Public Member Functions

- static void [createSegmentPathNetwork](#) (const stdair::BomRoot &)

#### 25.49.1 Detailed Description

Class handling the generation / instantiation of the network BOM.

Definition at line 34 of file [SegmentPathGenerator.hpp](#).

## 25.49.2 Member Function Documentation

### 25.49.2.1 void AIRTSP::SegmentPathGenerator::createSegmentPathNetwork (const stdair::BomRoot & iBomRoot) [static]

Generate the segment path network.

Definition at line 26 of file [SegmentPathGenerator.cpp](#).

Referenced by [AIRTSP::AIRTSP\\_Service::buildComplementaryLinks\(\)](#).

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

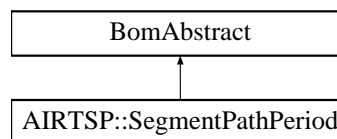
- [airtsp/command/SegmentPathGenerator.hpp](#)
- [airtsp/command/SegmentPathGenerator.cpp](#)

## 25.50 AIRTSP::SegmentPathPeriod Class Reference

Class representing a segment/path.

`#include <airtsp/bom/SegmentPathPeriod.hpp>`  
 AIRTSP::SegmentPathPeriod::

diagram for



### Public Types

- typedef [SegmentPathPeriodKey](#) Key\_T

### Public Member Functions

- const [Key\\_T](#) & [getKey](#) () const
- stdair::BomAbstract \*const [getParent](#) () const
- const stdair::PeriodStruct & [getDeparturePeriod](#) () const
- const [DateOffsetList\\_T](#) & [getBoardingDateOffsetList](#) () const
- const stdair::NbOfSegments\_T [getNbOfSegments](#) () const
- const stdair::NbOfAirlines\_T & [getNbOfAirlines](#) () const
- const stdair::Duration\_T & [getElapsedTime](#) () const
- const stdair::Duration\_T & [getBoardingTime](#) () const
- const stdair::HolderMap\_T & [getHolderMap](#) () const
- stdair::SegmentPeriod \* [getLastSegmentPeriod](#) () const
- stdair::SegmentPeriod \* [getFirstSegmentPeriod](#) () const
- const stdair::AirportCode\_T & [getDestination](#) () const
- [Key\\_T](#) [connectWithAnotherSegment](#) (const [SegmentPathPeriod](#) &) const
- bool [checkCircle](#) (const stdair::AirportCode\_T &) const
- bool [isAirlineFlown](#) (const stdair::AirlineCode\_T &) const
- bool [isDepartureDateValid](#) (const stdair::Date\_T &) const

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

### Protected Member Functions

- [SegmentPathPeriod](#) (const [Key\\_T](#) &)
- [~SegmentPathPeriod](#) ()

### Protected Attributes

- [Key\\_T \\_key](#)
- stdair::BomAbstract \* [\\_parent](#)
- stdair::HolderMap\_T [\\_holderMap](#)

### Friends

- class [stdair::FacBom](#)
- class [stdair::FacBomManager](#)
- class [boost::serialization::access](#)

## 25.50.1 Detailed Description

Class representing a segment/path. It corresponds to an actual travel solution from the origin to the destination, that is, a path that a traveller can take with actual scheduled flights.

Definition at line 39 of file [SegmentPathPeriod.hpp](#).

## 25.50.2 Member Typedef Documentation

### 25.50.2.1 typedef SegmentPathPeriodKey AIRTSP::SegmentPathPeriod::Key\_T

Definition allowing to retrieve the associated BOM key type.

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

## 25.50.3 Constructor & Destructor Documentation

### 25.50.3.1 AIRTSP::SegmentPathPeriod::SegmentPathPeriod (const Key\_T & iKey) [protected]

Main constructor.

Definition at line 43 of file [SegmentPathPeriod.cpp](#).

**25.50.3.2 AIRTSP::SegmentPathPeriod::~~SegmentPathPeriod () [protected]**

Destructor.

Definition at line 48 of file [SegmentPathPeriod.cpp](#).**25.50.4 Member Function Documentation****25.50.4.1 const Key\_T& AIRTSP::SegmentPathPeriod::getKey () const [inline]**

Get the primary key (destination airport).

Definition at line 60 of file [SegmentPathPeriod.hpp](#).References [\\_key](#).**25.50.4.2 stdair::BomAbstract\* const AIRTSP::SegmentPathPeriod::getParent () const [inline]**Get the parent (i.e., [OriginDestinationSet](#)) object.Definition at line 67 of file [SegmentPathPeriod.hpp](#).References [\\_parent](#).**25.50.4.3 const stdair::PeriodStruct& AIRTSP::SegmentPathPeriod::getDeparturePeriod () const [inline]**

Get the departure period (part of the primary key).

Definition at line 72 of file [SegmentPathPeriod.hpp](#).References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getPeriod\(\)](#).Referenced by [connectWithAnotherSegment\(\)](#), and [isDepartureDateValid\(\)](#).**25.50.4.4 const DateOffsetList\_T& AIRTSP::SegmentPathPeriod::getBoardingDateOffsetList () const [inline]**

Get the boarding date offset list (part of the primary key).

Definition at line 77 of file [SegmentPathPeriod.hpp](#).References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getBoardingDateOffsetList\(\)](#).Referenced by [connectWithAnotherSegment\(\)](#).**25.50.4.5 const stdair::NbOfSegments\_T AIRTSP::SegmentPathPeriod::getNbOfSegments () const [inline]**

Get the number of segments (part of the primary key).

Definition at line 82 of file [SegmentPathPeriod.hpp](#).References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getNbOfSegments\(\)](#).Referenced by [connectWithAnotherSegment\(\)](#).

**25.50.4.6    const stdair::NbOfAirlines\_T& AIRTSP::SegmentPathPeriod::getNbOfAirlines () const [inline]**

Get the number of airlines (part of the primary key).

Definition at line 87 of file [SegmentPathPeriod.hpp](#).

References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getNbOfAirlines\(\)](#).

**25.50.4.7    const stdair::Duration\_T& AIRTSP::SegmentPathPeriod::getElapsedTime () const [inline]**

Get the elapsed time (part of the primary key).

Definition at line 92 of file [SegmentPathPeriod.hpp](#).

References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getElapsedTime\(\)](#).

Referenced by [connectWithAnotherSegment\(\)](#).

**25.50.4.8    const stdair::Duration\_T& AIRTSP::SegmentPathPeriod::getBoardingTime () const [inline]**

Get the boarding time (part of the primary key).

Definition at line 97 of file [SegmentPathPeriod.hpp](#).

References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::getBoardingTime\(\)](#).

Referenced by [connectWithAnotherSegment\(\)](#).

**25.50.4.9    const stdair::HolderMap\_T& AIRTSP::SegmentPathPeriod::getHolderMap () const [inline]**

Get the map of children holders (SegmentPeriod objects).

Definition at line 104 of file [SegmentPathPeriod.hpp](#).

References [\\_holderMap](#).

**25.50.4.10    stdair::SegmentPeriod \* AIRTSP::SegmentPathPeriod::getLastSegmentPeriod () const**

Get the last SegmentPeriod object of the list.

Return a NULL pointer if the list is empty.

Definition at line 91 of file [SegmentPathPeriod.cpp](#).

Referenced by [connectWithAnotherSegment\(\)](#), and [getDestination\(\)](#).

**25.50.4.11    stdair::SegmentPeriod \* AIRTSP::SegmentPathPeriod::getFirstSegmentPeriod () const**

Get the first SegmentPeriod object of the list.

Return a NULL pointer if the list is empty.

Definition at line 109 of file [SegmentPathPeriod.cpp](#).

Referenced by [connectWithAnotherSegment\(\)](#).

**25.50.4.12 const stdair::AirportCode\_T & AIRTSP::SegmentPathPeriod::getDestination () const**

Get the destination of the segment path (i.e., the destination of the last segment.

Definition at line 127 of file [SegmentPathPeriod.cpp](#).

References [getLastSegmentPeriod\(\)](#).

**25.50.4.13 SegmentPathPeriodKey AIRTSP::SegmentPathPeriod::connectWithAnotherSegment (const SegmentPathPeriod & iSingleSegmentPath) const**

Check whether the (i-1)-length segment path period can be merged with the single segment path period in order to create an i-length segment path period. The function will return a valid or non-valid segment path period key.

The two segment path period above can be fused (and will produce a valid new segment path period key) if:

1. A passenger can connect from the last segment of the first segment path and the first segment of the next segment path. These two segments should not create another segment.
2. There is no circle within the new segment path.
3. The intersection of the two periods is non-empty.

Definition at line 163 of file [SegmentPathPeriod.cpp](#).

References [checkCircle\(\)](#), [getBoardingDateOffsetList\(\)](#), [getBoardingTime\(\)](#), [getDeparturePeriod\(\)](#), [getElapsedTime\(\)](#), [getFirstSegmentPeriod\(\)](#), [getLastSegmentPeriod\(\)](#), [getNbOfSegments\(\)](#), [AIRTSP::SegmentPathPeriodKey::setBoardingDateOffsetList\(\)](#), [AIRTSP::SegmentPathPeriodKey::setBoardingTime\(\)](#), [AIRTSP::SegmentPathPeriodKey::setElapsedTime\(\)](#), and [AIRTSP::SegmentPathPeriodKey::setPeriod\(\)](#).

**25.50.4.14 bool AIRTSP::SegmentPathPeriod::checkCircle (const stdair::AirportCode\_T &) const**

Check whether the given destination airport is also the departure point of one of the other segment members. If yes, a circle exists.

Referenced by [connectWithAnotherSegment\(\)](#).

**25.50.4.15 bool AIRTSP::SegmentPathPeriod::isAirlineFlown (const stdair::AirlineCode\_T & iAirlineCode) const**

State whether or not the given airline is flown by (at least) one of the segments of the internal list.

Definition at line 135 of file [SegmentPathPeriod.cpp](#).

**25.50.4.16 bool AIRTSP::SegmentPathPeriod::isDepartureDateValid (const stdair::Date\_T & iDepartureDate) const**

Check whether the given departure date is included in the departure period of the segment path.

Definition at line 308 of file [SegmentPathPeriod.cpp](#).

References [getDeparturePeriod\(\)](#).

**25.50.4.17** `void AIRTSP::SegmentPathPeriod::toStream (std::ostream & ioOut) const`  
**[inline]**

Dump a Business Object into an output stream.

**Parameters:**

*ostream&* the output stream.

Definition at line 176 of file [SegmentPathPeriod.hpp](#).

References [toString\(\)](#).

**25.50.4.18** `void AIRTSP::SegmentPathPeriod::fromStream (std::istream & ioIn)` **[inline]**

Read a Business Object from an input stream.

**Parameters:**

*istream&* the input stream.

Definition at line 185 of file [SegmentPathPeriod.hpp](#).

**25.50.4.19** `std::string AIRTSP::SegmentPathPeriod::toString () const`

Get the serialised version of the Business Object.

Definition at line 52 of file [SegmentPathPeriod.cpp](#).

References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::toString\(\)](#).

Referenced by [toStream\(\)](#).

**25.50.4.20** `const std::string AIRTSP::SegmentPathPeriod::describeKey () const` **[inline]**

Get a string describing the key.

Definition at line 196 of file [SegmentPathPeriod.hpp](#).

References [\\_key](#), and [AIRTSP::SegmentPathPeriodKey::toString\(\)](#).

**25.50.4.21** `template<class Archive > void AIRTSP::SegmentPathPeriod::serialize (Archive & ar,  
const unsigned int iFileVersion)` **[inline]**

Serialisation.

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

References [\\_key](#).

## 25.50.5 Friends And Related Function Documentation

**25.50.5.1** `friend class stdair::FacBom` **[friend]**

Friend classes.

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

### 25.50.5.2 friend class stdair::FacBomManager [friend]

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

### 25.50.5.3 friend class boost::serialization::access [friend]

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

## 25.50.6 Member Data Documentation

### 25.50.6.1 Key\_T AIRTSP::SegmentPathPeriod::\_key [protected]

Primary key (segment/path characteristics: scheduled period, number of segments, number of airlines, elapsed time, boarding time).

Definition at line 249 of file [SegmentPathPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingDateOffsetList\(\)](#), [getBoardingTime\(\)](#), [getDeparturePeriod\(\)](#), [getElapsedTime\(\)](#), [getKey\(\)](#), [getNbOfAirlines\(\)](#), [getNbOfSegments\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

### 25.50.6.2 stdair::BomAbstract\* AIRTSP::SegmentPathPeriod::\_parent [protected]

Pointer on the parent ([OriginDestinationSet](#)) object.

Definition at line 254 of file [SegmentPathPeriod.hpp](#).

Referenced by [getParent\(\)](#).

### 25.50.6.3 stdair::HolderMap\_T AIRTSP::SegmentPathPeriod::\_holderMap [protected]

Map holding the children (SegmentPeriod objects).

#### Note:

The SegmentPeriod objects themselves have for parent the FlightPeriod class (not the [SegmentPathPeriod](#) class).

Definition at line 262 of file [SegmentPathPeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

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

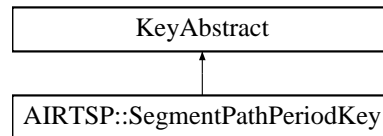
- [airtsp/bom/SegmentPathPeriod.hpp](#)
- [airtsp/bom/SegmentPathPeriod.cpp](#)

## 25.51 AIRTSP::SegmentPathPeriodKey Struct Reference

Structure representing the key of a segment/path.



#include <airtsp/bom/SegmentPathPeriodKey.hpp> Inheritance diagram for AIRTSP::SegmentPathPeriodKey::



### Public Member Functions

- [SegmentPathPeriodKey](#) (const stdair::PeriodStruct &, const stdair::Duration\_T &iBoardingTime, const stdair::Duration\_T &iElapsed, const [DateOffsetList\\_T](#) &, const stdair::NbOfAirlines\_T &)
- [SegmentPathPeriodKey](#) ()
- [SegmentPathPeriodKey](#) (const [SegmentPathPeriodKey](#) &)
- [~SegmentPathPeriodKey](#) ()
- const stdair::PeriodStruct & [getPeriod](#) () const
- const [DateOffsetList\\_T](#) & [getBoardingDateOffsetList](#) () const
- const stdair::NbOfSegments\_T [getNbOfSegments](#) () const
- const stdair::NbOfAirlines\_T & [getNbOfAirlines](#) () const
- const stdair::Duration\_T & [getElapsedTime](#) () const
- const stdair::Duration\_T & [getBoardingTime](#) () const
- void [setPeriod](#) (const stdair::PeriodStruct &iPeriod)
- void [setBoardingDateOffsetList](#) (const [DateOffsetList\\_T](#) &iList)
- void [setNbOfAirlines](#) (const stdair::NbOfAirlines\_T &iNbOfAirlines)
- void [setElapsedTime](#) (const stdair::Duration\_T &iElapsed)
- void [setBoardingTime](#) (const stdair::Duration\_T &iBoardingTime)
- const bool [isValid](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

### Friends

- class [boost::serialization::access](#)

#### 25.51.1 Detailed Description

Structure representing the key of a segment/path. That key specifies a travel solution from a geographical point (origin airport) to another (destination airport).

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

## 25.51.2 Constructor & Destructor Documentation

**25.51.2.1** AIRTSP::SegmentPathPeriodKey::SegmentPathPeriodKey (const stdair::PeriodStruct & *iPeriod*, const stdair::Duration\_T & *iBoardingTime*, const stdair::Duration\_T & *iElapsed*, const DateOffsetList\_T & *iBoardingDateOffsetList*, const stdair::NbOfAirlines\_T & *iNbOfAirlines*)

Constructor.

Definition at line 40 of file [SegmentPathPeriodKey.cpp](#).

**25.51.2.2** AIRTSP::SegmentPathPeriodKey::SegmentPathPeriodKey ()

Default constructor.

Definition at line 22 of file [SegmentPathPeriodKey.cpp](#).

**25.51.2.3** AIRTSP::SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey & *iSPPK*)

Copy constructor.

Definition at line 30 of file [SegmentPathPeriodKey.cpp](#).

**25.51.2.4** AIRTSP::SegmentPathPeriodKey::~~SegmentPathPeriodKey ()

Destructor.

Definition at line 53 of file [SegmentPathPeriodKey.cpp](#).

## 25.51.3 Member Function Documentation

**25.51.3.1** const stdair::PeriodStruct& AIRTSP::SegmentPathPeriodKey::getPeriod () const  
[inline]

Get the active days-of-week.

Definition at line 68 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::getDeparturePeriod\(\)](#).

**25.51.3.2** const DateOffsetList\_T& AIRTSP::SegmentPathPeriodKey::getBoardingDateOffsetList () const [inline]

Get the list of boarding date off-sets.

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

Referenced by [AIRTSP::SegmentPathPeriod::getBoardingDateOffsetList\(\)](#).

**25.51.3.3** const stdair::NbOfSegments\_T AIRTSP::SegmentPathPeriodKey::getNbOfSegments () const [inline]

Get the number of segments.

Definition at line 82 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::getNbOfSegments\(\)](#).

**25.51.3.4** `const stdair::NbOfAirlines_T& AIRTSP::SegmentPathPeriodKey::getNbOfAirlines ()  
const [inline]`

Get the number of airlines.

Definition at line 89 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::getNbOfAirlines\(\)](#).

**25.51.3.5** `const stdair::Duration_T& AIRTSP::SegmentPathPeriodKey::getElapsedTime () const  
[inline]`

Get the elapsed time.

Definition at line 96 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::getElapsedTime\(\)](#).

**25.51.3.6** `const stdair::Duration_T& AIRTSP::SegmentPathPeriodKey::getBoardingTime ()  
const [inline]`

Get the boarding time.

Definition at line 103 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::getBoardingTime\(\)](#).

**25.51.3.7** `void AIRTSP::SegmentPathPeriodKey::setPeriod (const stdair::PeriodStruct & iPeriod)  
[inline]`

Set the active days-of-week.

Definition at line 111 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

**25.51.3.8** `void AIRTSP::SegmentPathPeriodKey::setBoardingDateOffsetList (const  
DateOffsetList_T & iList) [inline]`

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

Referenced by [AIRTSP::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

**25.51.3.9** `void AIRTSP::SegmentPathPeriodKey::setNbOfAirlines (const stdair::NbOfAirlines_T  
& iNbOfAirlines) [inline]`

Set the number of airlines.

Definition at line 120 of file [SegmentPathPeriodKey.hpp](#).

**25.51.3.10** void AIRTSP::SegmentPathPeriodKey::setElapsedTime (const stdair::Duration\_T & *iElapsed*) [inline]

Set the elapsed time.

Definition at line 125 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

**25.51.3.11** void AIRTSP::SegmentPathPeriodKey::setBoardingTime (const stdair::Duration\_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 130 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRTSP::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

**25.51.3.12** const bool AIRTSP::SegmentPathPeriodKey::isValid () const [inline]

Check if the key is valid (i.e. the departure period is valid).

Definition at line 138 of file [SegmentPathPeriodKey.hpp](#).

**25.51.3.13** void AIRTSP::SegmentPathPeriodKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

**Parameters:**

*ostream&* the output stream.

Definition at line 57 of file [SegmentPathPeriodKey.cpp](#).

References [toString\(\)](#).

**25.51.3.14** void AIRTSP::SegmentPathPeriodKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

**Parameters:**

*istream&* the input stream.

Definition at line 62 of file [SegmentPathPeriodKey.cpp](#).

**25.51.3.15** const std::string AIRTSP::SegmentPathPeriodKey::toString () const

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

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

Referenced by [AIRTSP::SegmentPathPeriod::describeKey\(\)](#), [toStream\(\)](#), and [AIRTSP::SegmentPathPeriod::toString\(\)](#).

**25.51.3.16** `template<class Archive > void AIRTSP::SegmentPathPeriodKey::serialize (Archive & ar, const unsigned int iFileVersion) [inline]`

Serialisation.

Definition at line 98 of file [SegmentPathPeriodKey.cpp](#).

## 25.51.4 Friends And Related Function Documentation

**25.51.4.1** `friend class boost::serialization::access [friend]`

Definition at line 34 of file [SegmentPathPeriodKey.hpp](#).

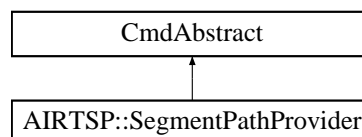
The documentation for this struct was generated from the following files:

- [airtsp/bom/SegmentPathPeriodKey.hpp](#)
- [airtsp/bom/SegmentPathPeriodKey.cpp](#)

## 25.52 AIRTSP::SegmentPathProvider Class Reference

Class building the travel solutions from airline schedules.

`#include <airtsp/command/SegmentPathProvider.hpp>`Inheritance diagram for AIRTSP::SegmentPathProvider::



### Friends

- class [AIRTSP\\_Service](#)

### 25.52.1 Detailed Description

Class building the travel solutions from airline schedules.

Definition at line 27 of file [SegmentPathProvider.hpp](#).

## 25.52.2 Friends And Related Function Documentation

### 25.52.2.1 friend class AIRTSP\_Service [friend]

Definition at line 28 of file [SegmentPathProvider.hpp](#).

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

- [airtsp/command/SegmentPathProvider.hpp](#)
- [airtsp/command/SegmentPathProvider.cpp](#)

## 25.53 AIRTSP::SegmentPeriodHelper Class Reference

Class representing the actual business functions for an airline segment-period.

```
#include <airtsp/bom/SegmentPeriodHelper.hpp>
```

### Static Public Member Functions

- static void [fill](#) (stdair::SegmentPeriod &, const [SegmentStruct](#) &)
- static void [fill](#) (stdair::SegmentPeriod &, const [LegStructList\\_T](#) &)

### 25.53.1 Detailed Description

Class representing the actual business functions for an airline segment-period.

Definition at line 22 of file [SegmentPeriodHelper.hpp](#).

### 25.53.2 Member Function Documentation

#### 25.53.2.1 void AIRTSP::SegmentPeriodHelper::fill (stdair::SegmentPeriod & *ioSegmentPeriod*, const [SegmentStruct](#) & *iSegmentStruct*) [static]

Fill the attributes of the given segment-period with the cabins and classes.

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

References [AIRTSP::SegmentCabinStruct::\\_cabinCode](#), [AIRTSP::SegmentStruct::\\_cabinList](#), and [AIRTSP::SegmentCabinStruct::\\_classes](#).

#### 25.53.2.2 void AIRTSP::SegmentPeriodHelper::fill (stdair::SegmentPeriod & *ioSegmentPeriod*, const [LegStructList\\_T](#) & *iLegList*) [static]

Fill the attributes of the given segment-period with the list of used legs.

Definition at line 29 of file [SegmentPeriodHelper.cpp](#).

References [AIRTSP::LegStruct::\\_boardingPoint](#), [AIRTSP::LegStruct::\\_offDateOffset](#), and [AIRTSP::LegStruct::\\_offTime](#).

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

- [airtsp/bom/SegmentPeriodHelper.hpp](#)

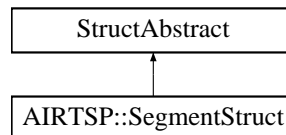
- [airtsp/bom/SegmentPeriodHelper.cpp](#)

## 25.54 AIRTSP::SegmentStruct Struct Reference

`#include <airtsp/bom/SegmentStruct.hpp>`  
 AIRTSP::SegmentStruct::Inheritance

diagram

for



### Public Member Functions

- void [fill](#) (stdair::SegmentDate &) const
- const std::string [describe](#) () const

### Public Attributes

- stdair::AirportCode\_T [\\_boardingPoint](#)
- stdair::Date\_T [\\_boardingDate](#)
- stdair::Duration\_T [\\_boardingTime](#)
- stdair::AirportCode\_T [\\_offPoint](#)
- stdair::Date\_T [\\_offDate](#)
- stdair::Duration\_T [\\_offTime](#)
- stdair::Duration\_T [\\_elapsed](#)
- [SegmentCabinStructList\\_T\\_cabinList](#)

#### 25.54.1 Detailed Description

Utility Structure for the parsing of Segment structures.

Definition at line 24 of file [SegmentStruct.hpp](#).

#### 25.54.2 Member Function Documentation

##### 25.54.2.1 void AIRTSP::SegmentStruct::fill (stdair::SegmentDate & *ioSegmentDate*) const

Fill the SegmentDate objects with the attributes of the [SegmentStruct](#).

Definition at line 35 of file [SegmentStruct.cpp](#).

##### 25.54.2.2 const std::string AIRTSP::SegmentStruct::describe () const

Give a description of the structure (for display purposes).

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

References [\\_boardingPoint](#), [\\_boardingTime](#), [\\_cabinList](#), [\\_elapsed](#), [\\_offPoint](#), [\\_offTime](#), and [AIRTSP::SegmentCabinStruct::describe\(\)](#).

Referenced by [AIRTSP::FlightPeriodStruct::describe\(\)](#).

### 25.54.3 Member Data Documentation

#### 25.54.3.1 `stdair::AirportCode_T AIRTSP::SegmentStruct::_boardingPoint`

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

Referenced by [AIRTSP::FlightPeriodStruct::addFareFamily\(\)](#), [AIRTSP::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRTSP::FlightPeriodStruct::buildSegments\(\)](#), [describe\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#).

#### 25.54.3.2 `stdair::Date_T AIRTSP::SegmentStruct::_boardingDate`

Definition at line 27 of file [SegmentStruct.hpp](#).

#### 25.54.3.3 `stdair::Duration_T AIRTSP::SegmentStruct::_boardingTime`

Definition at line 28 of file [SegmentStruct.hpp](#).

Referenced by [describe\(\)](#).

#### 25.54.3.4 `stdair::AirportCode_T AIRTSP::SegmentStruct::_offPoint`

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

Referenced by [AIRTSP::FlightPeriodStruct::addFareFamily\(\)](#), [AIRTSP::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRTSP::FlightPeriodStruct::buildSegments\(\)](#), [describe\(\)](#), and [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#).

#### 25.54.3.5 `stdair::Date_T AIRTSP::SegmentStruct::_offDate`

Definition at line 30 of file [SegmentStruct.hpp](#).

#### 25.54.3.6 `stdair::Duration_T AIRTSP::SegmentStruct::_offTime`

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

Referenced by [describe\(\)](#).



**25.54.3.7 stdair::Duration\_T AIRTSP::SegmentStruct::\_elapsed**

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

Referenced by [describe\(\)](#).

**25.54.3.8 SegmentCabinStructList\_T AIRTSP::SegmentStruct::\_cabinList**

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

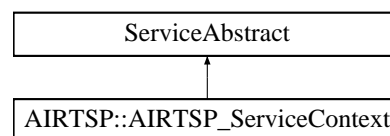
Referenced by [AIRTSP::FlightPeriodStruct::addFareFamily\(\)](#), [AIRTSP::FlightPeriodStruct::addSegmentCabin\(\)](#), [describe\(\)](#), and [AIRTSP::SegmentPeriodHelper::fill\(\)](#).

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

- [airtsp/bom/SegmentStruct.hpp](#)
- [airtsp/bom/SegmentStruct.cpp](#)

**25.55 ServiceAbstract Class Reference**

Inheritance diagram for ServiceAbstract::



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

- [airtsp/service/AIRTSP\\_ServiceContext.hpp](#)

**25.56 AIRTSP::ServiceAbstract Class Reference**

```
#include <airtsp/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](#) ()

### 25.56.1 Detailed Description

Base class for the Service layer.

Definition at line 14 of file [ServiceAbstract.hpp](#).

### 25.56.2 Constructor & Destructor Documentation

#### 25.56.2.1 virtual AIRTSP::ServiceAbstract::~ServiceAbstract() [inline, virtual]

Destructor.

Definition at line 18 of file [ServiceAbstract.hpp](#).

#### 25.56.2.2 AIRTSP::ServiceAbstract::ServiceAbstract() [inline, protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 30 of file [ServiceAbstract.hpp](#).

### 25.56.3 Member Function Documentation

#### 25.56.3.1 virtual void AIRTSP::ServiceAbstract::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

##### Parameters:

*ostream&* the output stream.

Definition at line 22 of file [ServiceAbstract.hpp](#).

#### 25.56.3.2 virtual void AIRTSP::ServiceAbstract::fromStream (std::istream & ioIn) [inline, virtual]

Read a Business Object from an input stream.

##### Parameters:

*istream&* the input stream.

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

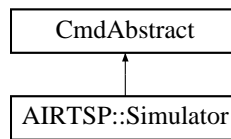
Referenced by [operator>>\(\)](#).

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

- [airtsp/service/ServiceAbstract.hpp](#)

## 25.57 AIRTSP::Simulator Class Reference

`#include <airtsp/command/Simulator.hpp>` Inheritance diagram for AIRTSP::Simulator::



### Static Public Member Functions

- static void [simulate](#) (stdair::BomRoot &)

#### 25.57.1 Detailed Description

Class implementing a small simulation, which uses the Airline Schedule.

Definition at line 18 of file [Simulator.hpp](#).

#### 25.57.2 Member Function Documentation

##### 25.57.2.1 void AIRTSP::Simulator::simulate (stdair::BomRoot & *ioBomRoot*) [static]

Perform a small simulation, which uses the Airline Schedule.

#### Parameters:

*stdair::BomRoot&* Root of the BOM tree.

Definition at line 19 of file [Simulator.cpp](#).

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

- airtsp/command/[Simulator.hpp](#)
- airtsp/command/[Simulator.cpp](#)

## 25.58 airtsp::store\_adult\_passenger\_type Struct Reference

### Public Member Functions

- [store\\_adult\\_passenger\\_type](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

#### 25.58.1 Detailed Description

Store the parsed passenger type.

Definition at line 145 of file [BookingRequestParser.cpp](#).

## 25.58.2 Constructor & Destructor Documentation

### 25.58.2.1 airtsp::store\_adult\_passenger\_type::store\_adult\_passenger\_type (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 147 of file [BookingRequestParser.cpp](#).

## 25.58.3 Member Function Documentation

### 25.58.3.1 void airtsp::store\_adult\_passenger\_type::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse adult passenger type.

Definition at line 151 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchString\\_T::\\_passengerList](#), [\\_searchString](#), [airtsp::SearchString\\_T::\\_tmpPassenger](#), [airtsp::Passenger\\_T::\\_type](#), and [airtsp::Passenger\\_T::ADULT](#).

## 25.58.4 Member Data Documentation

### 25.58.4.1 SearchString\_T& airtsp::store\_adult\_passenger\_type::\_searchString

Definition at line 160 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.59 airtsp::store\_airline\_code Struct Reference

### Public Member Functions

- [store\\_airline\\_code](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.59.1 Detailed Description

Store the parsed airline code.

Definition at line 92 of file [BookingRequestParser.cpp](#).

## 25.59.2 Constructor & Destructor Documentation

### 25.59.2.1 airtsp::store\_airline\_code::store\_airline\_code (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 94 of file [BookingRequestParser.cpp](#).

## 25.59.3 Member Function Documentation

### 25.59.3.1 void airtsp::store\_airline\_code::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse the airline code.

Definition at line 98 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchString\\_T::\\_airlineList](#), [airtsp::Airline\\_T::\\_code](#), [\\_searchString](#), and [airtsp::SearchString\\_T::\\_tmpAirline](#).

## 25.59.4 Member Data Documentation

### 25.59.4.1 SearchString\_T& airtsp::store\_airline\_code::\_searchString

Definition at line 107 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.60 airtsp::store\_airline\_name Struct Reference

### Public Member Functions

- [store\\_airline\\_name](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.60.1 Detailed Description

Store the parsed airline name.

Definition at line 111 of file [BookingRequestParser.cpp](#).

## 25.60.2 Constructor & Destructor Documentation

### 25.60.2.1 airtsp::store\_airline\_name::store\_airline\_name (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 113 of file [BookingRequestParser.cpp](#).

## 25.60.3 Member Function Documentation

### 25.60.3.1 void airtsp::store\_airline\_name::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse the airline name.

Definition at line 117 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchString\\_T::\\_airlineList](#), [airtsp::Airline\\_T::\\_name](#), [\\_searchString](#), and [airtsp::SearchString\\_T::\\_tmpAirline](#).

## 25.60.4 Member Data Documentation

### 25.60.4.1 SearchString\_T& airtsp::store\_airline\_name::\_searchString

Definition at line 126 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.61 airtsp::store\_airline\_sign Struct Reference

### Public Member Functions

- [store\\_airline\\_sign](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (bool iAirlineSign) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.61.1 Detailed Description

Store the airline sign (+/-).

Definition at line 77 of file [BookingRequestParser.cpp](#).

### 25.61.2 Constructor & Destructor Documentation

#### 25.61.2.1 airtsp::store\_airline\_sign::store\_airline\_sign (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 79 of file [BookingRequestParser.cpp](#).

### 25.61.3 Member Function Documentation

#### 25.61.3.1 void airtsp::store\_airline\_sign::operator() (bool iAirlineSign) const [inline]

Parse the airline sign.

Definition at line 83 of file [BookingRequestParser.cpp](#).

References [airtsp::Airline\\_T::\\_isPreferred](#), [\\_searchString](#), and [airtsp::SearchString\\_T::\\_tmpAirline](#).

### 25.61.4 Member Data Documentation

#### 25.61.4.1 SearchString\_T& airtsp::store\_airline\_sign::\_searchString

Definition at line 88 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.62 airtsp::store\_child\_passenger\_type Struct Reference

### Public Member Functions

- [store\\_child\\_passenger\\_type](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.62.1 Detailed Description

Store the parsed passenger type.

Definition at line 164 of file [BookingRequestParser.cpp](#).

### 25.62.2 Constructor & Destructor Documentation

#### 25.62.2.1 airtsp::store\_child\_passenger\_type::store\_child\_passenger\_type (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 166 of file [BookingRequestParser.cpp](#).

### 25.62.3 Member Function Documentation

#### 25.62.3.1 void airtsp::store\_child\_passenger\_type::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse child passenger type.

Definition at line 170 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchString\\_T::\\_passengerList](#), [\\_searchString](#), [airtsp::SearchString\\_T::\\_tmpPassenger](#), [airtsp::Passenger\\_T::\\_type](#), and [airtsp::Passenger\\_T::CHILD](#).

### 25.62.4 Member Data Documentation

#### 25.62.4.1 SearchString\_T& airtsp::store\_child\_passenger\_type::\_searchString

Definition at line 179 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.63 airtsp::store\_date Struct Reference

### Public Member Functions

- [store\\_date](#) ([SearchString\\_T](#) &ioSearchString)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.63.1 Detailed Description

Store a parsed date.

Definition at line 58 of file [BookingRequestParser.cpp](#).



## 25.63.2 Constructor & Destructor Documentation

### 25.63.2.1 airtsp::store\_date::store\_date (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 60 of file [BookingRequestParser.cpp](#).

## 25.63.3 Member Function Documentation

### 25.63.3.1 void airtsp::store\_date::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse the date.

Definition at line 64 of file [BookingRequestParser.cpp](#).

References [airtsp::Date\\_T::\\_date](#), [airtsp::SearchString\\_T::\\_dateList](#), [\\_searchString](#), [airtsp::SearchString\\_T::\\_tmpDate](#), and [airtsp::Date\\_T::getDate\(\)](#).

## 25.63.4 Member Data Documentation

### 25.63.4.1 SearchString\_T& airtsp::store\_date::\_searchString

Definition at line 73 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.64 airtsp::store\_passenger\_number Struct Reference

### Public Member Functions

- [store\\_passenger\\_number](#) ([SearchString\\_T](#) &ioSearchString)
- void [operator\(\)](#) (unsigned int iNumber) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.64.1 Detailed Description

Store the parsed number of passengers.

Definition at line 130 of file [BookingRequestParser.cpp](#).

### 25.64.2 Constructor & Destructor Documentation

#### 25.64.2.1 airtsp::store\_passenger\_number::store\_passenger\_number (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 132 of file [BookingRequestParser.cpp](#).

### 25.64.3 Member Function Documentation

#### 25.64.3.1 void airtsp::store\_passenger\_number::operator() (unsigned int iNumber) const [inline]

Parse number of passengers.

Definition at line 136 of file [BookingRequestParser.cpp](#).

References [airtsp::Passenger\\_T::\\_number](#), [\\_searchString](#), and [airtsp::SearchString\\_T::\\_tmpPassenger](#).

### 25.64.4 Member Data Documentation

#### 25.64.4.1 SearchString\_T& airtsp::store\_passenger\_number::\_searchString

Definition at line 141 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.65 airtsp::store\_pet\_passenger\_type Struct Reference

### Public Member Functions

- [store\\_pet\\_passenger\\_type](#) (SearchString\_T &ioSearchString)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.65.1 Detailed Description

Store the parsed passenger type.

Definition at line 183 of file [BookingRequestParser.cpp](#).

## 25.65.2 Constructor & Destructor Documentation

### 25.65.2.1 airtsp::store\_pet\_passenger\_type::store\_pet\_passenger\_type (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 185 of file [BookingRequestParser.cpp](#).

## 25.65.3 Member Function Documentation

### 25.65.3.1 void airtsp::store\_pet\_passenger\_type::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse pet passenger type.

Definition at line 189 of file [BookingRequestParser.cpp](#).

References [airtsp::SearchString\\_T::\\_passengerList](#), [\\_searchString](#), [airtsp::SearchString\\_T::\\_tmpPassenger](#), [airtsp::Passenger\\_T::\\_type](#), and [airtsp::Passenger\\_T::PET](#).

## 25.65.4 Member Data Documentation

### 25.65.4.1 SearchString\_T& airtsp::store\_pet\_passenger\_type::\_searchString

Definition at line 198 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.66 airtsp::store\_place\_element Struct Reference

### Public Member Functions

- [store\\_place\\_element](#) ([SearchString\\_T](#) &ioSearchString)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [SearchString\\_T](#) & [\\_searchString](#)

### 25.66.1 Detailed Description

Store the parsed place element.

Definition at line 37 of file [BookingRequestParser.cpp](#).

## 25.66.2 Constructor & Destructor Documentation

### 25.66.2.1 airtsp::store\_place\_element::store\_place\_element (SearchString\_T & ioSearchString) [inline]

Constructor.

Definition at line 39 of file [BookingRequestParser.cpp](#).

## 25.66.3 Member Function Documentation

### 25.66.3.1 void airtsp::store\_place\_element::operator() (iterator\_t iStr, iterator\_t iStrEnd) const [inline]

Parse the place.

Definition at line 43 of file [BookingRequestParser.cpp](#).

References [airtsp::Place\\_T::\\_name](#), [\\_searchString](#), and [airtsp::SearchString\\_T::\\_tmpPlace](#).

## 25.66.4 Member Data Documentation

### 25.66.4.1 SearchString\_T& airtsp::store\_place\_element::\_searchString

Definition at line 54 of file [BookingRequestParser.cpp](#).

Referenced by [operator\(\)](#).

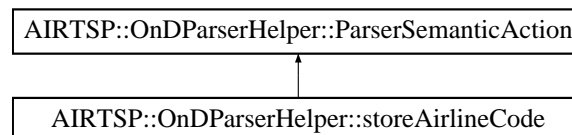
The documentation for this struct was generated from the following file:

- [airtsp/batches/BookingRequestParser.cpp](#)

## 25.67 AIRTSP::OnDParserHelper::storeAirlineCode Struct Reference

```
#include <airtsp/command/OnDParserHelper.hpp>
```

Inheritance diagram for AIRTSP::OnDParserHelper::storeAirlineCode:



## Public Member Functions

- [storeAirlineCode](#) (OnDPeriodStruct &)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

## Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

### 25.67.1 Detailed Description

Store the parsed airline code.

Definition at line 90 of file [OnDParserHelper.hpp](#).

### 25.67.2 Constructor & Destructor Documentation

#### 25.67.2.1 AIRTSP::OnDParserHelper::storeAirlineCode::storeAirlineCode (OnDPeriodStruct & ioOnDPeriod)

Actor Constructor.

Definition at line 139 of file [OnDParserHelper.cpp](#).

### 25.67.3 Member Function Documentation

#### 25.67.3.1 void AIRTSP::OnDParserHelper::storeAirlineCode::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 144 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_airlineCode](#), [AIRTSP::OnDPeriodStruct::\\_airlineCodeList](#), [AIRTSP::OnDPeriodStruct::\\_nbOfAirlines](#), and [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#).

### 25.67.4 Member Data Documentation

#### 25.67.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

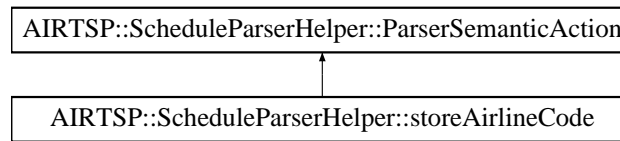
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.68 AIRTSP::ScheduleParserHelper::storeAirlineCode Struct Reference

```
#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for
AIRTSP::ScheduleParserHelper::storeAirlineCode::
```



## Public Member Functions

- [storeAirlineCode](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

## Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.68.1 Detailed Description

Store the parsed airline code.

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

### 25.68.2 Constructor & Destructor Documentation

#### 25.68.2.1 AIRTSP::ScheduleParserHelper::storeAirlineCode::storeAirlineCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 32 of file [ScheduleParserHelper.cpp](#).

### 25.68.3 Member Function Documentation

#### 25.68.3.1 void AIRTSP::ScheduleParserHelper::storeAirlineCode::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 37 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_airlineCode](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_legList](#).

### 25.68.4 Member Data Documentation

#### 25.68.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

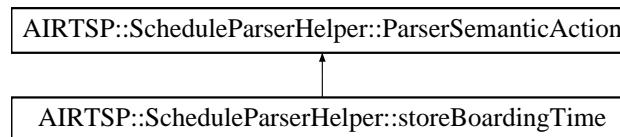
Referenced by AIRTSP::ScheduleParserHelper::doEndFlight::operator(), AIRTSP::ScheduleParserHelper::storeFClasses::operator(), AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator(), AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator(), AIRTSP::ScheduleParserHelper::storeFamilyCode::operator(), AIRTSP::ScheduleParserHelper::storeClasses::operator(), AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator(), AIRTSP::ScheduleParserHelper::storeCapacity::operator(), AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeElapsedTime::operator(), AIRTSP::ScheduleParserHelper::storeOffTime::operator(), AIRTSP::ScheduleParserHelper::storeBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(), AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator(), AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(), AIRTSP::ScheduleParserHelper::storeFlightNumber::operator() and operator().

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.69 AIRTSP::ScheduleParserHelper::storeBoardingTime Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeBoardingTime:



### Public Member Functions

- storeBoardingTime (FlightPeriodStruct &)
- void operator() (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- FlightPeriodStruct & \_flightPeriod

#### 25.69.1 Detailed Description

Store the boarding time.

Definition at line 109 of file ScheduleParserHelper.hpp.

## 25.69.2 Constructor & Destructor Documentation

### 25.69.2.1 AIRTSP::ScheduleParserHelper::storeBoardingTime::storeBoardingTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 191 of file [ScheduleParserHelper.cpp](#).

## 25.69.3 Member Function Documentation

### 25.69.3.1 void AIRTSP::ScheduleParserHelper::storeBoardingTime::operator() (iterator\_t *iStr*, iterator\_t *iStrEnd*) const

Actor Function (functor).

Definition at line 196 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegStruct::\\_boardingTime](#), [AIRTSP::FlightPeriodStruct::\\_dateOffset](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::FlightPeriodStruct::\\_itSeconds](#), and [AIRTSP::FlightPeriodStruct::getTime\(\)](#).

## 25.69.4 Member Data Documentation

### 25.69.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

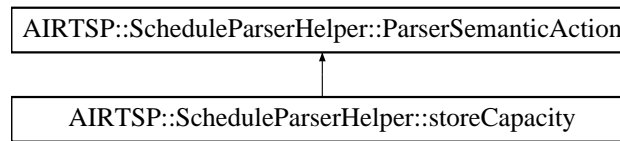
The documentation for this struct was generated from the following files:

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.70 AIRTSP::ScheduleParserHelper::storeCapacity Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeCapacity::





### Public Member Functions

- [storeCapacity](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) (double *iReal*) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.70.1 Detailed Description

Store the parsed capacity.

Definition at line 141 of file [ScheduleParserHelper.hpp](#).

### 25.70.2 Constructor & Destructor Documentation

#### 25.70.2.1 AIRTSP::ScheduleParserHelper::storeCapacity::storeCapacity ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 263 of file [ScheduleParserHelper.cpp](#).

### 25.70.3 Member Function Documentation

#### 25.70.3.1 void AIRTSP::ScheduleParserHelper::storeCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 268 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegStruct::\\_cabinList](#), [AIRTSP::LegCabinStruct::\\_capacity](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), and [AIRTSP::FlightPeriodStruct::\\_itLegCabin](#).

### 25.70.4 Member Data Documentation

#### 25.70.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

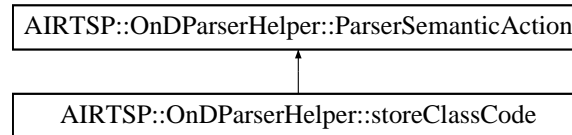
Referenced by AIRTSP::ScheduleParserHelper::doEndFlight::operator(), AIRTSP::ScheduleParserHelper::storeFClasses::operator(), AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator(), AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator(), AIRTSP::ScheduleParserHelper::storeFamilyCode::operator(), AIRTSP::ScheduleParserHelper::storeClasses::operator(), AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator(), AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeElapsedTime::operator(), AIRTSP::ScheduleParserHelper::storeOffTime::operator(), AIRTSP::ScheduleParserHelper::storeBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(), AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator(), AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(), AIRTSP::ScheduleParserHelper::storeFlightNumber::operator(), and AIRTSP::ScheduleParserHelper::storeAirlineCode::operator().

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.71 AIRTSP::OnDParserHelper::storeClassCode Struct Reference

#include <airtsp/command/OnDParserHelper.hpp> Inheritance diagram for AIRTSP::OnDParserHelper::storeClassCode::



### Public Member Functions

- storeClassCode (OnDPeriodStruct &)
- void operator() (char iChar) const

### Public Attributes

- OnDPeriodStruct & \_onDPeriod

#### 25.71.1 Detailed Description

Store the parsed class code.

Definition at line 98 of file OnDParserHelper.hpp.

### 25.71.2 Constructor & Destructor Documentation

#### 25.71.2.1 AIRTSP::OnDParserHelper::storeClassCode::storeClassCode (OnDPeriodStruct & ioOnDPeriod)

Actor Constructor.

Definition at line 172 of file [OnDParserHelper.cpp](#).

### 25.71.3 Member Function Documentation

#### 25.71.3.1 void AIRTSP::OnDParserHelper::storeClassCode::operator() (char iChar) const

Actor Function (functor).

Definition at line 177 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_classCode](#), [AIRTSP::OnDPeriodStruct::\\_classCodeList](#), and [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#).

### 25.71.4 Member Data Documentation

#### 25.71.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

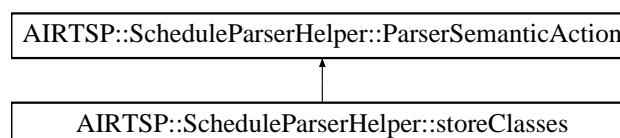
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.72 AIRTSP::ScheduleParserHelper::storeClasses Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>`Inheritance diagram for AIRTSP::ScheduleParserHelper::storeClasses::



**Public Member Functions**

- [storeClasses](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

**Public Attributes**

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

**25.72.1 Detailed Description**

Store the parsed list of class codes.

Definition at line 184 of file [ScheduleParserHelper.hpp](#).

**25.72.2 Constructor & Destructor Documentation****25.72.2.1 AIRTSP::ScheduleParserHelper::storeClasses::storeClasses ([FlightPeriodStruct](#) & *ioFlightPeriod*)**

Actor Constructor.

Definition at line 345 of file [ScheduleParserHelper.cpp](#).

**25.72.3 Member Function Documentation****25.72.3.1 void AIRTSP::ScheduleParserHelper::storeClasses::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const**

Actor Function (functor).

Definition at line 350 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_areSegmentDefinitionsSpecific](#), [AIRTSP::SegmentCabinStruct::\\_classes](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itSegment](#), [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#), and [AIRTSP::FlightPeriodStruct::addSegmentCabin\(\)](#).

**25.72.4 Member Data Documentation****25.72.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]**

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoarding::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#)

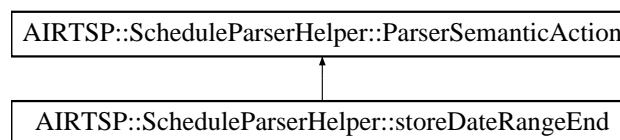
AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeElapsedTime::operator(), AIRTSP::ScheduleParserHelper::storeOffTime::operator(), AIRTSP::ScheduleParserHelper::storeBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(), AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator(), AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(), AIRTSP::ScheduleParserHelper::storeFlightNumber::operator() and AIRTSP::ScheduleParserHelper::storeAirlineCode::operator().

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.73 AIRTSP::ScheduleParserHelper::storeDateRangeEnd Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeDateRangeEnd:



### Public Member Functions

- storeDateRangeEnd (FlightPeriodStruct &)
- void operator() (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- FlightPeriodStruct & \_flightPeriod

#### 25.73.1 Detailed Description

Store the end of the date range.

Definition at line 61 of file [ScheduleParserHelper.hpp](#).

#### 25.73.2 Constructor & Destructor Documentation

##### 25.73.2.1 AIRTSP::ScheduleParserHelper::storeDateRangeEnd::storeDateRangeEnd (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 77 of file [ScheduleParserHelper.cpp](#).

### 25.73.3 Member Function Documentation

#### 25.73.3.1 void AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 82 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_airlineCode](#), [AIRTSP::LegStruct::\\_airlineCode](#), [AIRTSP::FlightPeriodStruct::\\_dateRange](#), [AIRTSP::FlightPeriodStruct::\\_dateRangeEnd](#), [AIRTSP::FlightPeriodStruct::\\_dateRangeStart](#), [AIRTSP::FlightPeriodStruct::\\_flightNumber](#), [AIRTSP::LegStruct::\\_flightNumber](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::FlightPeriodStruct::\\_itSeconds](#), and [AIRTSP::FlightPeriodStruct::getDate\(\)](#).

### 25.73.4 Member Data Documentation

#### 25.73.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

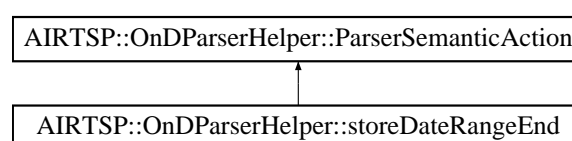
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.74 AIRTSP::OnDParserHelper::storeDateRangeEnd Struct Reference

`#include <airtsp/command/OnDParserHelper.hpp>` Inheritance diagram for AIRTSP::OnDParserHelper::storeDateRangeEnd:



**Public Member Functions**

- [storeDateRangeEnd](#) ([OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

**Public Attributes**

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

**25.74.1 Detailed Description**

Store the end of the date range.

Definition at line 66 of file [OnDParserHelper.hpp](#).

**25.74.2 Constructor & Destructor Documentation****25.74.2.1 AIRTSP::OnDParserHelper::storeDateRangeEnd::storeDateRangeEnd**  
([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

Definition at line 83 of file [OnDParserHelper.cpp](#).

**25.74.3 Member Function Documentation****25.74.3.1 void AIRTSP::OnDParserHelper::storeDateRangeEnd::operator()** ([iterator\\_t](#) *iStr*,  
[iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 88 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_datePeriod](#), [AIRTSP::OnDPeriodStruct::\\_dateRangeEnd](#), [AIRTSP::OnDPeriodStruct::\\_dateRangeStart](#), [AIRTSP::OnDPeriodStruct::\\_itSeconds](#), [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#), and [AIRTSP::OnDPeriodStruct::getDate\(\)](#).

**25.74.4 Member Data Documentation****25.74.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod**  
[inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

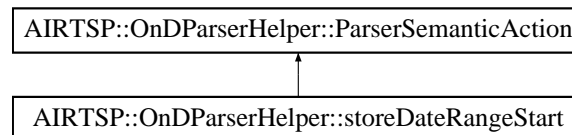
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.75 AIRTSP::OnDParserHelper::storeDateRangeStart Struct Reference

#include <[airtsp/command/OnDParserHelper.hpp](#)> Inheritance diagram for AIRTSP::OnDParserHelper::storeDateRangeStart::



### Public Member Functions

- [storeDateRangeStart](#) ([OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

### 25.75.1 Detailed Description

Store the start of the date range.

Definition at line 58 of file [OnDParserHelper.hpp](#).

### 25.75.2 Constructor & Destructor Documentation

#### 25.75.2.1 AIRTSP::OnDParserHelper::storeDateRangeStart::storeDateRangeStart ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

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

### 25.75.3 Member Function Documentation

#### 25.75.3.1 void AIRTSP::OnDParserHelper::storeDateRangeStart::operator() ([iterator\\_t](#) *iStr*, [iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 71 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_dateRangeStart](#), [AIRTSP::OnDPeriodStruct::\\_itSeconds](#), [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#), and [AIRTSP::OnDPeriodStruct::getDate\(\)](#).



### 25.75.4 Member Data Documentation

#### 25.75.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

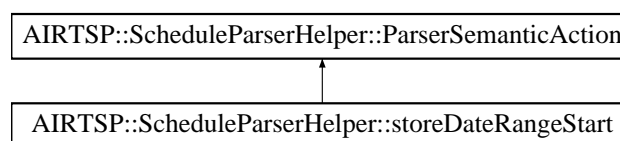
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [operator\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.76 AIRTSP::ScheduleParserHelper::storeDateRangeStart Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeDateRangeStart:



### Public Member Functions

- [storeDateRangeStart](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.76.1 Detailed Description

Store the start of the date range.

Definition at line 53 of file [ScheduleParserHelper.hpp](#).

### 25.76.2 Constructor & Destructor Documentation

#### 25.76.2.1 AIRTSP::ScheduleParserHelper::storeDateRangeStart::storeDateRangeStart ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 62 of file [ScheduleParserHelper.cpp](#).

### 25.76.3 Member Function Documentation

#### 25.76.3.1 void AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 67 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_dateRangeStart](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itSeconds](#), and [AIRTSP::FlightPeriodStruct::getDate\(\)](#).

### 25.76.4 Member Data Documentation

#### 25.76.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

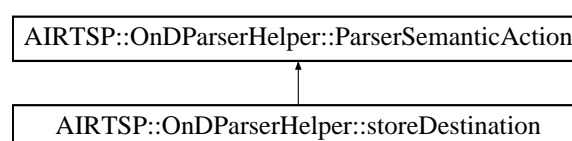
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.77 AIRTSP::OnDParserHelper::storeDestination Struct Reference

`#include <airtsp/command/OnDParserHelper.hpp>` Inheritance diagram for AIRTSP::OnDParserHelper::storeDestination::



**Public Member Functions**

- [storeDestination](#) ([OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

**Public Attributes**

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

**25.77.1 Detailed Description**

Store the parsed destination.

Definition at line 50 of file [OnDParserHelper.hpp](#).

**25.77.2 Constructor & Destructor Documentation****25.77.2.1 AIRTSP::OnDParserHelper::storeDestination::storeDestination ([OnDPeriodStruct](#) & *ioOnDPeriod*)**

Actor Constructor.

Definition at line 50 of file [OnDParserHelper.cpp](#).

**25.77.3 Member Function Documentation****25.77.3.1 void AIRTSP::OnDParserHelper::storeDestination::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const**

Actor Function (functor).

Definition at line 55 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_destination](#), and [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#).

**25.77.4 Member Data Documentation****25.77.4.1 [OnDPeriodStruct&](#) AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [*inherited*]**

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

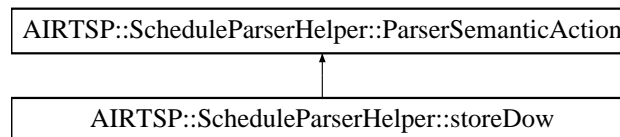
The documentation for this struct was generated from the following files:

- [airtsp/command/OnDParserHelper.hpp](#)

- [airtsp/command/OnDParserHelper.cpp](#)

## 25.78 AIRTSP::ScheduleParserHelper::storeDow Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeDow::



### Public Member Functions

- [storeDow](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.78.1 Detailed Description

Store the DOW (day of the Week).

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

### 25.78.2 Constructor & Destructor Documentation

#### 25.78.2.1 AIRTSP::ScheduleParserHelper::storeDow::storeDow ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 104 of file [ScheduleParserHelper.cpp](#).

### 25.78.3 Member Function Documentation

#### 25.78.3.1 void AIRTSP::ScheduleParserHelper::storeDow::operator() ([iterator\\_t](#) *iStr*, [iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 109 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_dow](#), and [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#).

## 25.78.4 Member Data Documentation

### 25.78.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

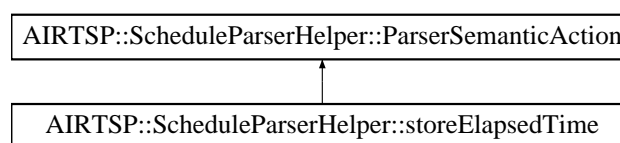
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#) and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.79 AIRTSP::ScheduleParserHelper::storeElapsedTime Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeElapsedTime::



### Public Member Functions

- [storeElapsedTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.79.1 Detailed Description

Store the elapsed time.

Definition at line 125 of file [ScheduleParserHelper.hpp](#).

## 25.79.2 Constructor & Destructor Documentation

### 25.79.2.1 AIRTSP::ScheduleParserHelper::storeElapsedTime::storeElapsedTime (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 230 of file [ScheduleParserHelper.cpp](#).

## 25.79.3 Member Function Documentation

### 25.79.3.1 void AIRTSP::ScheduleParserHelper::storeElapsedTime::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 235 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_dateOffset](#), [AIRTSP::LegStruct::\\_elapsed](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::FlightPeriodStruct::\\_itSeconds](#), [AIRTSP::LegStruct::\\_offDateOffset](#), and [AIRTSP::FlightPeriodStruct::getTime\(\)](#).

## 25.79.4 Member Data Documentation

### 25.79.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

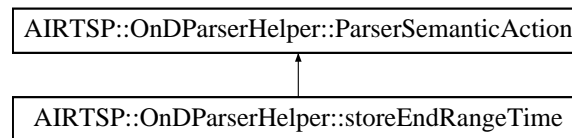
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingLeg::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.80 AIRTSP::OnDParserHelper::storeEndRangeTime Struct Reference

#include <airtsp/command/OnDParserHelper.hpp> Inheritance diagram for AIRTSP::OnDParserHelper::storeEndRangeTime::



### Public Member Functions

- [storeEndRangeTime](#) ([OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

### 25.80.1 Detailed Description

Store the end range time.

Definition at line 82 of file [OnDParserHelper.hpp](#).

### 25.80.2 Constructor & Destructor Documentation

#### 25.80.2.1 AIRTSP::OnDParserHelper::storeEndRangeTime::storeEndRangeTime ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

Definition at line 124 of file [OnDParserHelper.cpp](#).

### 25.80.3 Member Function Documentation

#### 25.80.3.1 void AIRTSP::OnDParserHelper::storeEndRangeTime::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 129 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_itSeconds](#), [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#), [AIRTSP::OnDPeriodStruct::\\_timeRangeEnd](#), and [AIRTSP::OnDPeriodStruct::getTime\(\)](#).

## 25.80.4 Member Data Documentation

### 25.80.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

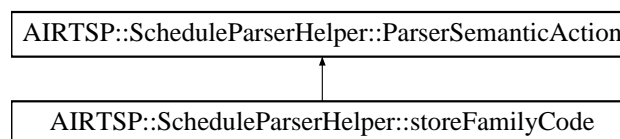
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.81 AIRTSP::ScheduleParserHelper::storeFamilyCode Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeFamilyCode::



### Public Member Functions

- [storeFamilyCode](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) (int iCode) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.81.1 Detailed Description

Store the parsed family code.

Definition at line 192 of file [ScheduleParserHelper.hpp](#).

### 25.81.2 Constructor & Destructor Documentation

#### 25.81.2.1 AIRTSP::ScheduleParserHelper::storeFamilyCode::storeFamilyCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.



Definition at line 370 of file [ScheduleParserHelper.cpp](#).

### 25.81.3 Member Function Documentation

#### 25.81.3.1 void AIRTSP::ScheduleParserHelper::storeFamilyCode::operator() (int iCode) const

Actor Function (functor).

Definition at line 375 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::SegmentCabinStruct::\\_itFamilyCode](#), and [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#).

### 25.81.4 Member Data Documentation

#### 25.81.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

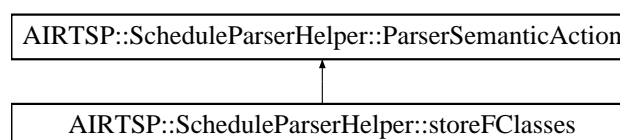
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.82 AIRTSP::ScheduleParserHelper::storeFClasses Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeFClasses:



**Public Member Functions**

- [storeFClasses](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

**Public Attributes**

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

**25.82.1 Detailed Description**

Store the parsed list of class codes (for families).

Definition at line 216 of file [ScheduleParserHelper.hpp](#).

**25.82.2 Constructor & Destructor Documentation****25.82.2.1 AIRTSP::ScheduleParserHelper::storeFClasses::storeFClasses ([FlightPeriodStruct](#) & *ioFlightPeriod*)**

Actor Constructor.

Definition at line 410 of file [ScheduleParserHelper.cpp](#).

**25.82.3 Member Function Documentation****25.82.3.1 void AIRTSP::ScheduleParserHelper::storeFClasses::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const**

Actor Function (functor).

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

References [AIRTSP::FlightPeriodStruct::\\_areSegmentDefinitionsSpecific](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::SegmentCabinStruct::\\_itFamilyCode](#), [AIRTSP::SegmentCabinStruct::\\_itFFDisutilityCurveKey](#), [AIRTSP::SegmentCabinStruct::\\_itFRAT5CurveKey](#), [AIRTSP::FlightPeriodStruct::\\_itSegment](#), [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#), and [AIRTSP::FlightPeriodStruct::addFareFamily\(\)](#).

**25.82.4 Member Data Documentation****25.82.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]**

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoarding](#)

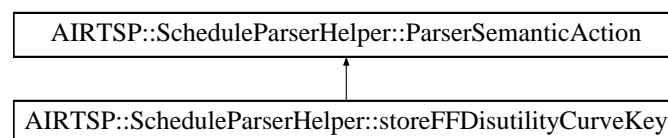
AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator(), AIRTSP::ScheduleParserHelper::storeCapacity::operator(), AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeElapsedTime::operator(), AIRTSP::ScheduleParserHelper::storeOffTime::operator(), AIRTSP::ScheduleParserHelper::storeBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(), AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator(), AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(), AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(), AIRTSP::ScheduleParserHelper::storeFlightNumber::operator() and AIRTSP::ScheduleParserHelper::storeAirlineCode::operator().

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.83 AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::



### Public Member Functions

- storeFFDisutilityCurveKey (FlightPeriodStruct &)
- void operator() (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- FlightPeriodStruct & \_flightPeriod

### 25.83.1 Detailed Description

Store the FFDisutility curve key.

Definition at line 208 of file [ScheduleParserHelper.hpp](#).

### 25.83.2 Constructor & Destructor Documentation

#### 25.83.2.1 AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::storeFFDisutilityCurveKey (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 397 of file [ScheduleParserHelper.cpp](#).

### 25.83.3 Member Function Documentation

#### 25.83.3.1 void AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 402 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::SegmentCabinStruct::\\_itFFDisutilityCurveKey](#), and [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#).

### 25.83.4 Member Data Documentation

#### 25.83.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

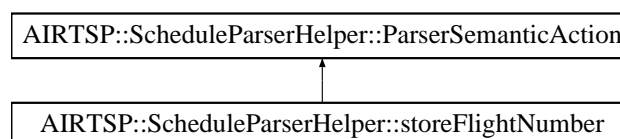
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingF::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.84 AIRTSP::ScheduleParserHelper::storeFlightNumber Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>`Inheritance diagram for AIRTSP::ScheduleParserHelper::storeFlightNumber:



### Public Member Functions

- [storeFlightNumber](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (unsigned int *iNumber*) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.84.1 Detailed Description

Store the parsed flight number.

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

#### 25.84.2 Constructor & Destructor Documentation

##### 25.84.2.1 AIRTSP::ScheduleParserHelper::storeFlightNumber::storeFlightNumber ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 50 of file [ScheduleParserHelper.cpp](#).

#### 25.84.3 Member Function Documentation

##### 25.84.3.1 void AIRTSP::ScheduleParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 55 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_flightNumber](#), and [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#).

#### 25.84.4 Member Data Documentation

##### 25.84.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#)

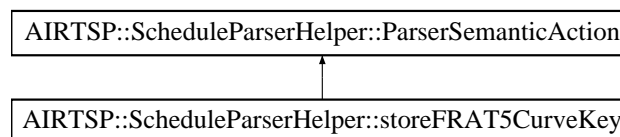
[AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [operator\(\)](#), and  
[AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.85 AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::



### Public Member Functions

- [storeFRAT5CurveKey](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.85.1 Detailed Description

Store the FRAT5 curve key.

Definition at line 200 of file [ScheduleParserHelper.hpp](#).

#### 25.85.2 Constructor & Destructor Documentation

##### 25.85.2.1 AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::storeFRAT5CurveKey ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 383 of file [ScheduleParserHelper.cpp](#).

### 25.85.3 Member Function Documentation

#### 25.85.3.1 void AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 388 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::SegmentCabinStruct::\\_itFRAT5CurveKey](#), and [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#).

### 25.85.4 Member Data Documentation

#### 25.85.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

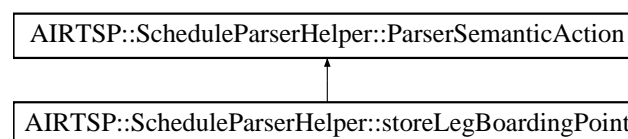
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.86 AIRTSP::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>`Inheritance diagram for AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::







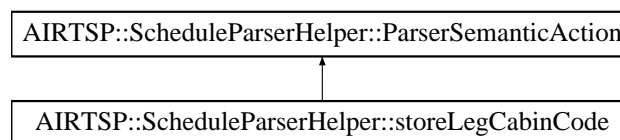
AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator(), AIRTSP::ScheduleParserHelper::storeElapsedTime::operator(), AIRTSP::ScheduleParserHelper::storeOffTime::operator(), AIRTSP::ScheduleParserHelper::storeBoardingTime::operator(), AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(), AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator(), AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(), operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(), AIRTSP::ScheduleParserHelper::storeFlightNumber::operator(), and AIRTSP::ScheduleParserHelper::storeAirlineCode::operator().

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.87 AIRTSP::ScheduleParserHelper::storeLegCabinCode Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeLegCabinCode::



### Public Member Functions

- storeLegCabinCode (FlightPeriodStruct &)
- void operator() (char iChar) const

### Public Attributes

- FlightPeriodStruct & \_flightPeriod

#### 25.87.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 133 of file [ScheduleParserHelper.hpp](#).

#### 25.87.2 Constructor & Destructor Documentation

##### 25.87.2.1 AIRTSP::ScheduleParserHelper::storeLegCabinCode::storeLegCabinCode (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 251 of file [ScheduleParserHelper.cpp](#).

### 25.87.3 Member Function Documentation

#### 25.87.3.1 void AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 256 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegCabinStruct::\\_cabinCode](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_itLegCabin](#).

### 25.87.4 Member Data Documentation

#### 25.87.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

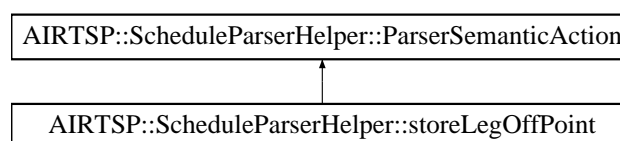
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.88 AIRTSP::ScheduleParserHelper::storeLegOffPoint Struct Reference

#include <[airtsp/command/ScheduleParserHelper.hpp](#)> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeLegOffPoint::



### Public Member Functions

- [storeLegOffPoint](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.88.1 Detailed Description

Store the parsed leg off point.

Definition at line 85 of file [ScheduleParserHelper.hpp](#).

#### 25.88.2 Constructor & Destructor Documentation

##### 25.88.2.1 AIRTSP::ScheduleParserHelper::storeLegOffPoint::storeLegOffPoint([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 146 of file [ScheduleParserHelper.cpp](#).

#### 25.88.3 Member Function Documentation

##### 25.88.3.1 void AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator() ([iterator\\_t](#) *iStr*, [iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 151 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::LegStruct::\\_offPoint](#), and [AIRTSP::FlightPeriodStruct::addAirport\(\)](#).

#### 25.88.4 Member Data Documentation

##### 25.88.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoarding::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), and [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#).

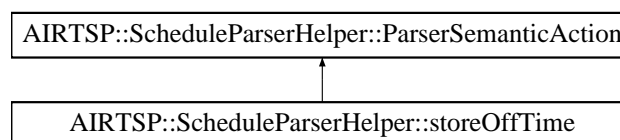
[AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#),  
[AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#)  
 and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.89 AIRTSP::ScheduleParserHelper::storeOffTime Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeOffTime:



### Public Member Functions

- [storeOffTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.89.1 Detailed Description

Store the off time.

Definition at line 117 of file [ScheduleParserHelper.hpp](#).

#### 25.89.2 Constructor & Destructor Documentation

##### 25.89.2.1 AIRTSP::ScheduleParserHelper::storeOffTime::storeOffTime ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 209 of file [ScheduleParserHelper.cpp](#).

### 25.89.3 Member Function Documentation

#### 25.89.3.1 void AIRTSP::ScheduleParserHelper::storeOffTime::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 214 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegStruct::\\_boardingDateOffset](#), [AIRTSP::FlightPeriodStruct::\\_dateOffset](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itLeg](#), [AIRTSP::FlightPeriodStruct::\\_itSeconds](#), [AIRTSP::LegStruct::\\_offTime](#), and [AIRTSP::FlightPeriodStruct::getTime\(\)](#).

### 25.89.4 Member Data Documentation

#### 25.89.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

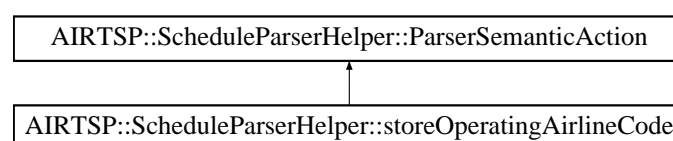
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.90 AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode:



## Public Member Functions

- [storeOperatingAirlineCode](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

## Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.90.1 Detailed Description

Store the parsed operating airline code.

Definition at line 93 of file [ScheduleParserHelper.hpp](#).

### 25.90.2 Constructor & Destructor Documentation

#### 25.90.2.1 AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::storeOperatingAirlineCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 162 of file [ScheduleParserHelper.cpp](#).

### 25.90.3 Member Function Documentation

#### 25.90.3.1 void AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 167 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegStruct::\\_airlineCode](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_itLeg](#).

### 25.90.4 Member Data Documentation

#### 25.90.4.1 [FlightPeriodStruct](#)& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#)

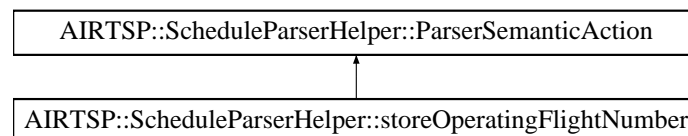
AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator(),  
 AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator(),  
 AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRTSP::ScheduleParserHelper::storeDow::operator(),  
 AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator(), AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator(),  
 AIRTSP::ScheduleParserHelper::storeFlightNumber::operator(), and AIRTSP::ScheduleParserHelper::storeAirlineCode::operator()

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

- airtsp/command/ScheduleParserHelper.hpp
- airtsp/command/ScheduleParserHelper.cpp

## 25.91 AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::



### Public Member Functions

- [storeOperatingFlightNumber](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (unsigned int iNumber) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.91.1 Detailed Description

Store the parsed operating flight number.

Definition at line 101 of file [ScheduleParserHelper.hpp](#).

### 25.91.2 Constructor & Destructor Documentation

#### 25.91.2.1 AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::storeOperatingFlightNumber (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 179 of file [ScheduleParserHelper.cpp](#).

### 25.91.3 Member Function Documentation

#### 25.91.3.1 void AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 184 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::LegStruct::\\_flightNumber](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_itLeg](#).

### 25.91.4 Member Data Documentation

#### 25.91.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

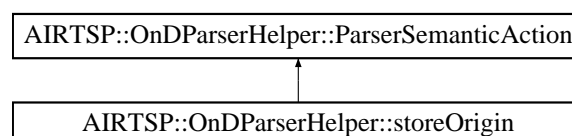
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingF::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.92 AIRTSP::OnDParserHelper::storeOrigin Struct Reference

`#include <airtsp/command/OnDParserHelper.hpp>` Inheritance diagram for AIRTSP::OnDParserHelper::storeOrigin:



### Public Member Functions

- [storeOrigin](#) ([OnDPeriodStruct](#) &)



- void [operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

## Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

### 25.92.1 Detailed Description

Store the parsed origin.

Definition at line 42 of file [OnDParserHelper.hpp](#).

### 25.92.2 Constructor & Destructor Documentation

#### 25.92.2.1 AIRTSP::OnDParserHelper::storeOrigin::storeOrigin ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

Definition at line 30 of file [OnDParserHelper.cpp](#).

### 25.92.3 Member Function Documentation

#### 25.92.3.1 void AIRTSP::OnDParserHelper::storeOrigin::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 35 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_airlineCode](#), [AIRTSP::OnDPeriodStruct::\\_airlineCodeList](#), [AIRTSP::OnDPeriodStruct::\\_classCode](#), [AIRTSP::OnDPeriodStruct::\\_classCodeList](#), [AIRTSP::OnDPeriodStruct::\\_nbOfAirlines](#), [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#), and [AIRTSP::OnDPeriodStruct::\\_origin](#).

### 25.92.4 Member Data Documentation

#### 25.92.4.1 [OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#) [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

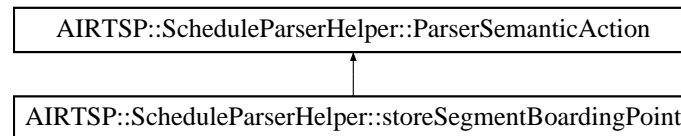
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)](#), and [operator\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.93 AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference

#include <airtsp/command/ScheduleParserHelper.hpp> Inheritance diagram for AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint:



### Public Member Functions

- [storeSegmentBoardingPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.93.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 160 of file [ScheduleParserHelper.hpp](#).

#### 25.93.2 Constructor & Destructor Documentation

##### 25.93.2.1 AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 308 of file [ScheduleParserHelper.cpp](#).

#### 25.93.3 Member Function Documentation

##### 25.93.3.1 void AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator() ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 313 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::SegmentStruct::\\_boardingPoint](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_itSegment](#).

### 25.93.4 Member Data Documentation

#### 25.93.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

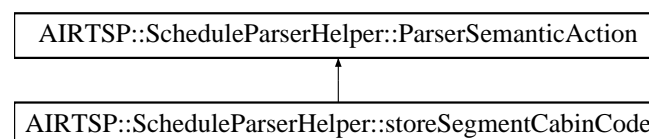
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#) and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.94 AIRTSP::ScheduleParserHelper::storeSegmentCabinCode Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeSegmentCabinCode:



### Public Member Functions

- [storeSegmentCabinCode](#) ([FlightPeriodStruct](#) &)
- [operator\(\)](#) ([char](#) iChar) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.94.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 176 of file [ScheduleParserHelper.hpp](#).

### 25.94.2 Constructor & Destructor Documentation

#### 25.94.2.1 AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::storeSegmentCabinCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 334 of file [ScheduleParserHelper.cpp](#).

### 25.94.3 Member Function Documentation

#### 25.94.3.1 void AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 339 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::SegmentCabinStruct::\\_cabinCode](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::\\_itSegmentCabin](#).

### 25.94.4 Member Data Documentation

#### 25.94.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

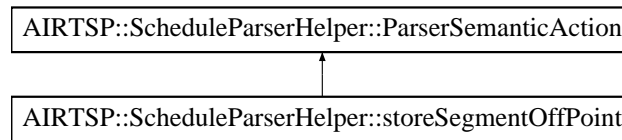
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.95 AIRTSP::ScheduleParserHelper::storeSegmentOffPoint Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::



### Public Member Functions

- [storeSegmentOffPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator\\_t](#) iStr, [iterator\\_t](#) iStrEnd) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

#### 25.95.1 Detailed Description

Store the parsed segment off point.

Definition at line 168 of file [ScheduleParserHelper.hpp](#).

#### 25.95.2 Constructor & Destructor Documentation

##### 25.95.2.1 AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::storeSegmentOffPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 321 of file [ScheduleParserHelper.cpp](#).

#### 25.95.3 Member Function Documentation

##### 25.95.3.1 void AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator() ([iterator\\_t](#) *iStr*, [iterator\\_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 326 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), [AIRTSP::FlightPeriodStruct::\\_itSegment](#), and [AIRTSP::SegmentStruct::\\_offPoint](#).

### 25.95.4 Member Data Documentation

#### 25.95.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

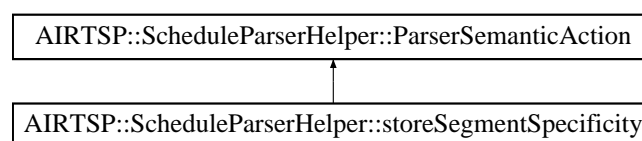
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFCClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)](#) and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.96 AIRTSP::ScheduleParserHelper::storeSegmentSpecificity Struct Reference

`#include <airtsp/command/ScheduleParserHelper.hpp>` Inheritance diagram for AIRTSP::ScheduleParserHelper::storeSegmentSpecificity:



### Public Member Functions

- [storeSegmentSpecificity](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (char iChar) const

### Public Attributes

- [FlightPeriodStruct](#) & [\\_flightPeriod](#)

### 25.96.1 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 152 of file [ScheduleParserHelper.hpp](#).

### 25.96.2 Constructor & Destructor Documentation

#### 25.96.2.1 AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::storeSegmentSpecificity (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 282 of file [ScheduleParserHelper.cpp](#).

### 25.96.3 Member Function Documentation

#### 25.96.3.1 void AIRTSP::ScheduleParserHelper::storeSegmentSpecificity::operator() (char iChar) const

Actor Function (functor).

Definition at line 287 of file [ScheduleParserHelper.cpp](#).

References [AIRTSP::FlightPeriodStruct::\\_airportList](#), [AIRTSP::FlightPeriodStruct::\\_airportOrderedList](#), [AIRTSP::FlightPeriodStruct::\\_areSegmentDefinitionsSpecific](#), [AIRTSP::ScheduleParserHelper::ParserSemanticAction::\\_flightPeriod](#), and [AIRTSP::FlightPeriodStruct::buildSegments\(\)](#).

### 25.96.4 Member Data Documentation

#### 25.96.4.1 FlightPeriodStruct& AIRTSP::ScheduleParserHelper::ParserSemanticAction::\_flightPeriod [inherited]

Actor Context.

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

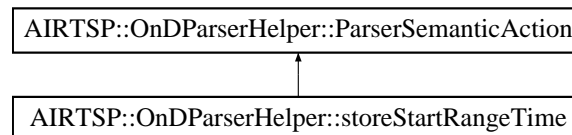
Referenced by [AIRTSP::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeSegmentBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegOffPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRTSP::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

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

- [airtsp/command/ScheduleParserHelper.hpp](#)
- [airtsp/command/ScheduleParserHelper.cpp](#)

## 25.97 AIRTSP::OnDParserHelper::storeStartRangeTime Struct Reference

`#include <airtsp/command/OnDParserHelper.hpp>` **Inheritance** **diagram** **for**  
AIRTSP::OnDParserHelper::storeStartRangeTime::



### Public Member Functions

- [storeStartRangeTime](#) (OnDPeriodStruct &)
- void [operator\(\)](#) (iterator\_t iStr, iterator\_t iStrEnd) const

### Public Attributes

- [OnDPeriodStruct](#) & [\\_onDPeriod](#)

#### 25.97.1 Detailed Description

Store the start range time.

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

#### 25.97.2 Constructor & Destructor Documentation

##### 25.97.2.1 AIRTSP::OnDParserHelper::storeStartRangeTime::storeStartRangeTime (OnDPeriodStruct & ioOnDPeriod)

Actor Constructor.

Definition at line 109 of file [OnDParserHelper.cpp](#).

#### 25.97.3 Member Function Documentation

##### 25.97.3.1 void AIRTSP::OnDParserHelper::storeStartRangeTime::operator() (iterator\_t iStr, iterator\_t iStrEnd) const

Actor Function (functor).

Definition at line 114 of file [OnDParserHelper.cpp](#).

References [AIRTSP::OnDPeriodStruct::\\_itSeconds](#), [AIRTSP::OnDParserHelper::ParserSemanticAction::\\_onDPeriod](#), [AIRTSP::OnDPeriodStruct::\\_timeRangeStart](#), and [AIRTSP::OnDPeriodStruct::getTime\(\)](#).



## 25.97.4 Member Data Documentation

### 25.97.4.1 OnDPeriodStruct& AIRTSP::OnDParserHelper::ParserSemanticAction::\_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

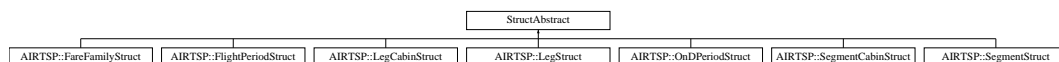
Referenced by [AIRTSP::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRTSP::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRTSP::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

- [airtsp/command/OnDParserHelper.hpp](#)
- [airtsp/command/OnDParserHelper.cpp](#)

## 25.98 StructAbstract Class Reference

Inheritance diagram for StructAbstract::

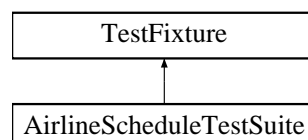


The documentation for this class was generated from the following files:

- [airtsp/bom/SegmentCabinStruct.hpp](#)
- [airtsp/bom/LegStruct.hpp](#)
- [airtsp/bom/FareFamilyStruct.hpp](#)
- [airtsp/bom/FlightPeriodStruct.hpp](#)
- [airtsp/bom/LegCabinStruct.hpp](#)
- [airtsp/bom/OnDPeriodStruct.hpp](#)
- [airtsp/bom/SegmentStruct.hpp](#)

## 25.99 TestFixture Class Reference

Inheritance diagram for TestFixture::



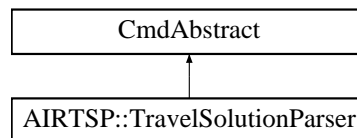
The documentation for this class was generated from the following file:

- [test/airtsp/AirlineScheduleTestSuite.hpp](#)

## 25.100 AIRTSP::TravelSolutionParser Class Reference

Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file.

#include <airtsp/command/TravelSolutionParser.hpp> Inheritance diagram for AIRTSP::TravelSolutionParser::



### Static Public Member Functions

- static bool [parseInputFileAndBuildBom](#) (const stdair::Filename\_T &)

#### 25.100.1 Detailed Description

Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file.

Definition at line 19 of file [TravelSolutionParser.hpp](#).

#### 25.100.2 Member Function Documentation

##### 25.100.2.1 static bool AIRTSP::TravelSolutionParser::parseInputFileAndBuildBom (const stdair::Filename\_T &) [static]

Parse the input values from a CSV-formatted travel solution file.

#### Parameters:

**const** std::string& iInputFileName Travel solution file to be parsed.

#### Returns:

bool Whether or not the parsing was successful.

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

- airtsp/command/[TravelSolutionParser.hpp](#)

## 26 File Documentation

### 26.1 airtsp/AIRTSP\_Service.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>

```

```
#include <stdair/stdair_file.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

### Classes

- class [AIRTSP::AIRTSP\\_Service](#)  
*Interface for the Airtsp Services.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.2 AIRTSP\_Service.hpp

```

00001 #ifndef __AIRTSP_SVC_AIRTSP_SERVICE_HPP
00002 #define __AIRTSP_SVC_AIRTSP_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_service_types.hpp>
00010 #include <stdair/stdair_file.hpp>
00011 #include <stdair/bom/TravelSolutionTypes.hpp>
00012
00014 namespace stdair {
00015     class STDAIR_Service;
00016     class BomRoot;
00017     struct BasLogParams;
00018     struct BasDBParams;
00019     struct BookingRequestStruct;
00020     struct TravelSolutionStruct;
00021 }
00022
00023 namespace AIRTSP {
00024
00026     class AIRTSP_ServiceContext;
00027
00028
00032     class AIRTSP_Service {
00033     public:
00034         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00050         AIRTSP_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);
00051
00063         AIRTSP_Service (const stdair::BasLogParams&);
00064
00080         AIRTSP_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr);
00081
00090         void parseAndLoad (const stdair::ScheduleFilePath&);
00091
00101         void parseAndLoad (const stdair::ScheduleFilePath&,
00102                             const stdair::ODFilePath&);
00103
00107         ~AIRTSP_Service ();
00108
00109
00110     public:
00111         // ////////////////////////////////// Business Methods //////////////////////////////////
00119         void buildSampleBom();
00120
00124         void clonePersistentBom ();
00125
00129         void buildComplementaryLinks (stdair::BomRoot&);
00130
00135         void buildSegmentPathList (stdair::TravelSolutionList_T&,
00136                                     const stdair::BookingRequestStruct&);
00137
00143         void simulate();
00144
00145
00146     public:
00147         // ////////////////////////////////// Export support methods //////////////////////////////////
00159         std::string jsonExportFlightDateObjects (const stdair::AirlineCode_T&,
00160                                                 const stdair::FlightNumber_T&,
00161                                                 const stdair::Date_T& iDepartureDate
00162         ) const;
00163

```

```
00164 public:
00165     // //////////// Display support methods ////////////
00173     std::string csvDisplay() const;
00174
00188     std::string csvDisplay (const stdair::AirlineCode_T&,
00189                             const stdair::FlightNumber_T&,
00190                             const stdair::Date_T& iDepartureDate) const;
00191
00192 private:
00193     // ////////// Construction and Destruction helper methods //////////
00194     AIRTSP_Service();
00198
00199     AIRTSP_Service (const AIRTSP_Service&);
00203
00204     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00214                                                     const stdair::BasDBParams&);
00215
00216     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&);
00225
00226     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00235                             const bool iOwnStdairService);
00236
00237     void initServiceContext();
00242
00243     void initAirtspService();
00250
00251     void finalise();
00255
00256 private:
00257     // ////////// Service Context //////////
00258     AIRTSP_ServiceContext* _airtspServiceContext;
00263
00264 };
00265 }
00266 #endif // __AIRTSP_SVC_AIRTSP_SERVICE_HPP
```

## 26.3 airtsp/AIRTSP\_Types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_exceptions.hpp>
```

### Classes

- class [AIRTSP::SegmentDateNotFoundException](#)
- class [AIRTSP::OnDInputFileNotFoundException](#)
- class [AIRTSP::ScheduleInputFileNotFoundException](#)

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef boost::shared\_ptr< AIRTSP\_Service > [AIRTSP::AIRTSP\\_ServicePtr\\_T](#)

## 26.4 AIRTSP\_Types.hpp

```

00001 #ifndef __AIRTSP_AIRTSP_TYPES_HPP
00002 #define __AIRTSP_AIRTSP_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // Boost
00008 #include <boost/shared_ptr.hpp>
00009 // StdAir
00010 #include <stdair/stdair_exceptions.hpp>
00011
00012 namespace AIRTSP {
00013
00014     // Forward declarations
00015     class AIRTSP_Service;
00016
00017
00018     // ////////// Exceptions //////////
00023     class SegmentDateNotFoundException : public stdair::ParserException {
00024     public:
00028         SegmentDateNotFoundException (const std::string& iWhat)
00029             : stdair::ParserException (iWhat) {}
00030     };
00031
00035     class OnDInputFileNotFoundException : public stdair::FileNotFoundException {
00036     public:
00040         OnDInputFileNotFoundException (const std::string& iWhat)
00041             : stdair::FileNotFoundException (iWhat) {}
00042     };
00043
00047     class ScheduleInputFileNotFoundException
00048         : public stdair::FileNotFoundException {
00049     public:
00053         ScheduleInputFileNotFoundException (const std::string& iWhat)
00054             : stdair::FileNotFoundException (iWhat) {}
00055     };
00056
00057
00058     // ////////// Type definitions specific to Airtsp //////////
00062     typedef boost::shared_ptr<AIRTSP_Service> AIRTSP_ServicePtr_T;
00063
00064 }
00065 #endif // __AIRTSP_AIRTSP_TYPES_HPP

```

## 26.5 airtsp/basic/BasConst.cpp File Reference

```
#include <airtsp/basic/BasConst_General.hpp>
#include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- const std::duration<std::chrono::duration<std::chrono::seconds>>::duration\_type> [AIRTSP::MINIMUM\\_TIME\\_BETWEEN\\_REQUEST\\_AND\\_DEPARTURE](#) (4, 0, 0)

### Variables

- const int [AIRTSP::DEFAULT\\_NUMBER\\_OF\\_DRAWS\\_FOR\\_MC\\_SIMULATION](#) = 100000



## 26.6 BasConst.cpp

```
00001 #include <airtsp/basic/BasConst_General.hpp>
00002 #include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
00003
00004 namespace AIRTSP {
00005
00008     const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000;
00009
00012     const stdair::Duration_T MINIMUM_TIME_BETWEEN_REQUEST_AND_DEPARTURE (4, 0, 0);
00013
00014 }
```

## 26.7 airtsp/basic/BasConst\_AIRTSP\_Service.hpp File Reference

### Namespaces

- namespace [AIRTSP](#)

## 26.8 BasConst\_AIRTSP\_Service.hpp

```
00001 #ifndef __AIRTSP_BAS_BASCONST_AIRTSP_SERVICE_HPP
00002 #define __AIRTSP_BAS_BASCONST_AIRTSP_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace AIRTSP {
00009
00010 }
00011 #endif // __AIRTSP_BAS_BASCONST_AIRTSP_SERVICE_HPP
```

## 26.9 airtsp/basic/BasConst\_General.hpp File Reference

```
#include <stdair/stdair_date_time_types.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Variables

- const stdair::Duration\_T [AIRTSP::MINIMUM\\_TIME\\_BETWEEN\\_REQUEST\\_AND\\_DEPARTURE](#)

## 26.10 BasConst\_General.hpp

```
00001 #ifndef __AIRTSP_BAS_BASCONST_GENERAL_HPP
00002 #define __AIRTSP_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_date_time_types.hpp>
00009
00010 namespace AIRTSP {
00011
00014     extern const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION;
00015
00018     extern const stdair::Duration_T MINIMUM_TIME_BETWEEN_REQUEST_AND_DEPARTURE;
00019
00020 }
00021 #endif // __AIRTSP_BAS_BASCONST_GENERAL_HPP
```

## 26.11 airtsp/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 [AIRTSP](#)

### Typedefs

- typedef char [AIRTSP::char\\_t](#)
- typedef boost::spirit::classic::file\_iterator< [char\\_t](#) > [AIRTSP::iterator\\_t](#)
- typedef boost::spirit::classic::scanner< [iterator\\_t](#) > [AIRTSP::scanner\\_t](#)
- typedef boost::spirit::classic::rule< [scanner\\_t](#) > [AIRTSP::rule\\_t](#)
- typedef boost::spirit::classic::int\_parser< unsigned int, 10, 1, 1 > [AIRTSP::int1\\_p\\_t](#)
- typedef boost::spirit::classic::uint\_parser< unsigned int, 10, 2, 2 > [AIRTSP::uint2\\_p\\_t](#)
- typedef boost::spirit::classic::uint\_parser< unsigned int, 10, 4, 4 > [AIRTSP::uint4\\_p\\_t](#)
- typedef boost::spirit::classic::uint\_parser< unsigned int, 10, 1, 4 > [AIRTSP::uint1\\_4\\_p\\_t](#)
- typedef boost::spirit::classic::chset< [char\\_t](#) > [AIRTSP::chset\\_t](#)
- typedef boost::spirit::classic::impl::loop\_traits< [chset\\_t](#), unsigned int, unsigned int >::type [AIRTSP::repeat\\_p\\_t](#)
- typedef boost::spirit::classic::bounded< [uint2\\_p\\_t](#), unsigned int > [AIRTSP::bounded2\\_p\\_t](#)
- typedef boost::spirit::classic::bounded< [uint4\\_p\\_t](#), unsigned int > [AIRTSP::bounded4\\_p\\_t](#)
- typedef boost::spirit::classic::bounded< [uint1\\_4\\_p\\_t](#), unsigned int > [AIRTSP::bounded1\\_4\\_p\\_t](#)

## 26.12 BasParserTypes.hpp

```

00001 #ifndef __AIRTSP_BAS_BASCOMPARSERTYPES_HPP
00002 #define __AIRTSP_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/confix.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 AIRTSP {
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 // //////////////////////////////////////
00044 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00045
00046 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> uint2_p_t;
00047
00048 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t;
00049
00050 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
uint1_4_p_t;
00051
00052 typedef boost::spirit::classic::chset<char_t> chset_t;
00053
00054 typedef boost::spirit::classic::impl::loop_traits<chset_t,
unsigned int,
unsigned int>::type repeat_p_t;
00055
00056 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t;
00057 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t;
00058 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
bounded1_4_p_t;
00059
00060 }
00061 #endif // __AIRTSP_BAS_BASCOMPARSERTYPES_HPP

```

## 26.13 airtsp/batches/airtsp.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <fstream>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/AIRTSP_Service.hpp>
#include <airtsp/batches/BookingRequestParser.hpp>
#include <airtsp/config/airtsp-paths.hpp>
```

### Typedefs

- typedef std::vector< std::string > [WordList\\_T](#)

### Functions

- const std::string [K\\_AIRTSP\\_DEFAULT\\_LOG\\_FILENAME](#) ("airtsp.log")
- const std::string [K\\_AIRTSP\\_DEFAULT\\_INPUT\\_FILENAME](#) (STDAIR\_SAMPLE\_DIR"/schedule03.csv")
- const std::string [K\\_AIRTSP\\_DEFAULT\\_BOOKING\\_REQUEST](#) ("NCE BKK NCE 2007-04-21 2007-03-21 08:32:00 C 1 DF RO 5 NONE 10:00:00 2000.0 20.0")
- std::string [createStringFromWordList](#) (const [WordList\\_T](#) &iWordList)
- template<class T >  
std::ostream & [operator<<](#) (std::ostream &os, const std::vector< T > &v)
- int [readConfiguration](#) (int argc, char \*argv[], bool &ioIsBuiltin, bool &ioReadBookingRequestFromCmdLine, stdair::Filename\_T &ioInputFilename, std::string &ioLogFilename, std::string &ioBookingRequestString)
- stdair::BookingRequestStruct [parseBookingRequest](#) (const std::string &iRequestOption)
- int [main](#) (int argc, char \*argv[])

### Variables

- const bool [K\\_AIRTSP\\_DEFAULT\\_BUILT\\_IN\\_INPUT](#) = false
- const bool [K\\_AIRTSP\\_DEFAULT\\_BOOKING\\_REQUEST\\_MODE](#) = false



- `const int K_AIRTSP_EARLY_RETURN_STATUS = 99`

### 26.13.1 Typedef Documentation

#### 26.13.1.1 `typedef std::vector<std::string> WordList_T`

Definition at line 24 of file [airtsp.cpp](#).

### 26.13.2 Function Documentation

#### 26.13.2.1 `const std::string K_AIRTSP_DEFAULT_LOG_FILENAME ("airtsp.log")`

Default name and location for the log file.

Referenced by [readConfiguration\(\)](#).

#### 26.13.2.2 `const std::string K_AIRTSP_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR"/schedule03.csv")`

Default name and location for the (CSV) input file.

Referenced by [readConfiguration\(\)](#).

#### 26.13.2.3 `const std::string K_AIRTSP_DEFAULT_BOOKING_REQUEST ("NCE BKK NCE 2007-04-21 2007-03-21 08:32:00 C 1 DF RO 5 NONE 10:00:00 2000.0 20.0")`

Default booking request string, to be seached against the AirTSP network.

Referenced by [main\(\)](#).

#### 26.13.2.4 `std::string createStringFromWordList (const WordList_T & iWordList)`

Definition at line 59 of file [airtsp.cpp](#).

Referenced by [readConfiguration\(\)](#).

#### 26.13.2.5 `template<class T > std::ostream& operator<< (std::ostream & os, const std::vector< T > & v) [inline]`

Definition at line 77 of file [airtsp.cpp](#).

#### 26.13.2.6 `int readConfiguration (int argc, char * argv[], bool & ioIsBuiltin, bool & ioReadBookingRequestFromCmdLine, stdair::Filename_T & ioInputFilename, std::string & ioLogFilename, std::string & ioBookingRequestString)`

Read and parse the command line options.

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

References [createStringFromWordList\(\)](#), [K\\_AIRTSP\\_DEFAULT\\_BOOKING\\_REQUEST\\_MODE](#), [K\\_AIRTSP\\_DEFAULT\\_BUILT\\_IN\\_INPUT](#), [K\\_AIRTSP\\_DEFAULT\\_INPUT\\_FILENAME\(\)](#), [K\\_AIRTSP\\_DEFAULT\\_LOG\\_FILENAME\(\)](#), and [K\\_AIRTSP\\_EARLY\\_RETURN\\_STATUS](#).

Referenced by [main\(\)](#).

#### 26.13.2.7 stdair::BookingRequestStruct parseBookingRequest (const std::string & iRequestOption)

Definition at line 230 of file [airtsp.cpp](#).

Referenced by [main\(\)](#).

#### 26.13.2.8 int main (int argc, char \* argv[])

Definition at line 347 of file [airtsp.cpp](#).

References [AIRTSP::AIRTSP\\_Service::buildSampleBom\(\)](#), [AIRTSP::AIRTSP\\_Service::buildSegmentPathList\(\)](#), [K\\_AIRTSP\\_DEFAULT\\_BOOKING\\_REQUEST\(\)](#), [K\\_AIRTSP\\_EARLY\\_RETURN\\_STATUS](#), [AIRTSP::AIRTSP\\_Service::parseAndLoad\(\)](#), [parseBookingRequest\(\)](#), and [readConfiguration\(\)](#).

### 26.13.3 Variable Documentation

#### 26.13.3.1 const bool K\_AIRTSP\_DEFAULT\_BUILT\_IN\_INPUT = false

Default for the BOM tree building. The BOM tree can either be built-in or provided by an input file. That latter must then be given with the -s option.

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

Referenced by [readConfiguration\(\)](#).

#### 26.13.3.2 const bool K\_AIRTSP\_DEFAULT\_BOOKING\_REQUEST\_MODE = false

Default for the input type. It can be either built-in or provided by an input file. That latter must then be given with the -i option.

Definition at line 50 of file [airtsp.cpp](#).

Referenced by [readConfiguration\(\)](#).

#### 26.13.3.3 const int K\_AIRTSP\_EARLY\_RETURN\_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 84 of file [airtsp.cpp](#).

Referenced by [main\(\)](#), and [readConfiguration\(\)](#).

## 26.14 airtsp.cpp

```

00001 // STL
00002 #include <cassert>
00003 #include <sstream>
00004 #include <fstream>
00005 #include <string>
00006 // Boost (Extended STL)
00007 #include <boost/date_time/posix_time/posix_time.hpp>
00008 #include <boost/date_time/gregorian/gregorian.hpp>
00009 #include <boost/program_options.hpp>
00010 #include <boost/tokenizer.hpp>
00011 #include <boost/lexical_cast.hpp>
00012 // StdAir
00013 #include <stdair/STDAIR_Service.hpp>
00014 #include <stdair/bom/BomDisplay.hpp>
00015 #include <stdair/bom/BookingRequestStruct.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirTSP
00019 #include <airtsp/AIRTSP_Service.hpp>
00020 #include <airtsp/batches/BookingRequestParser.hpp>
00021 #include <airtsp/config/airtsp-paths.hpp>
00022
00023 // ////////// Type definitions //////////
00024 typedef std::vector<std::string> WordList_T;
00025
00026
00027 // ////////// Constants //////////
00031 const std::string K_AIRTSP_DEFAULT_LOG_FILENAME ("airtsp.log");
00032
00036 const std::string K_AIRTSP_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00037                                                    "/schedule03.csv");
00038
00044 const bool K_AIRTSP_DEFAULT_BUILT_IN_INPUT = false;
00045
00050 const bool K_AIRTSP_DEFAULT_BOOKING_REQUEST_MODE = false;
00051
00056 const std::string K_AIRTSP_DEFAULT_BOOKING_REQUEST ("NCE BKK NCE 2007-04-21 2007-
00057 03-21 08:32:00 C 1 DF RO 5 NONE 10:00:00 2000.0 20.0");
00058
00058 // //////////////////////////////////////
00059 std::string createStringFromWordList (const WordList_T& iWordList) {
00060     std::ostringstream oStr;
00061
00062     unsigned short idx = iWordList.size();
00063     for (WordList_T::const_iterator itWord = iWordList.begin();
00064          itWord != iWordList.end(); ++itWord, --idx) {
00065         const std::string& lWord = *itWord;
00066         oStr << lWord;
00067         if (idx > 1) {
00068             oStr << " ";
00069         }
00070     }
00071
00072     return oStr.str();
00073 }
00074
00075 // ////////// Parsing of Options & Configuration //////////
00076 // A helper function to simplify the main part.
00077 template<class T> std::ostream& operator<< (std::ostream& os,
00078                                           const std::vector<T>& v) {
00079     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00080     return os;
00081 }
00082
00084 const int K_AIRTSP_EARLY_RETURN_STATUS = 99;

```

```

00085
00087 int readConfiguration (int argc, char* argv[],
00088                          bool& ioIsBuiltin, bool& ioReadBookingRequestFromCmdLine,
00089                          stdair::Filename_T& ioInputFilename,
00090                          std::string& ioLogFilename,
00091                          std::string& ioBookingRequestString) {
00092
00093     // Default for the built-in input
00094     ioIsBuiltin = K_AIRTSP_DEFAULT_BUILT_IN_INPUT;
00095
00096     // Default for the booking request mode (whether it is read from command-line)
00097     ioReadBookingRequestFromCmdLine = K_AIRTSP_DEFAULT_BOOKING_REQUEST_MODE;
00098
00099     //
00100     WordList_T lWordList;
00101
00102     // Declare a group of options that will be allowed only on command line
00103     boost::program_options::options_description generic ("Generic options");
00104     generic.add_options()
00105         ("prefix", "print installation prefix")
00106         ("version,v", "print version string")
00107         ("help,h", "produce help message");
00108
00109     // Declare a group of options that will be allowed both on command
00110     // line and in config file
00111     boost::program_options::options_description config ("Configuration");
00112     config.add_options()
00113         ("builtin,b",
00114          "The sample BOM tree can be either built-in or parsed from input files. In t
00115          hat latter case, the -i/--input option must be specified as well")
00116         ("input,i",
00117          boost::program_options::value< std::string >(&ioInputFilename)->default_valu
00118          e(K_AIRTSP_DEFAULT_INPUT_FILENAME),
00119          "(CSV) input file specifying the schedule (flight-period) entries")
00120         ("log,l",
00121          boost::program_options::value< std::string >(&ioLogFilename)->default_value(
00122          K_AIRTSP_DEFAULT_LOG_FILENAME),
00123          "Filename for the logs")
00124         ("read_booking_request,r",
00125          "Indicates that a booking request is given as a command-line option. That la
00126          tter must then be given with the -b/--bkg_req option")
00127         ("bkg_req,q",
00128          boost::program_options::value< WordList_T >(&lWordList)->multitoken(),
00129          "Booking request word list (e.g. 'NCE BKK NCE 2007-04-21 2007-04-21 10:00:00
00130          C 1 DF RO 5 NONE 10:0:0 2000.0 20.0'), which should be located at the end of the
00131          command line (otherwise, the other options would be interpreted as part of that
00132          booking request word list)")
00133         ;
00134
00135     // Hidden options, will be allowed both on command line and
00136     // in config file, but will not be shown to the user.
00137     boost::program_options::options_description hidden ("Hidden options");
00138     hidden.add_options()
00139         ("copyright",
00140          boost::program_options::value< std::vector<std::string> >(),
00141          "Show the copyright (license)");
00142
00143     boost::program_options::options_description cmdline_options;
00144     cmdline_options.add(generic).add(config).add(hidden);
00145
00146     boost::program_options::options_description config_file_options;
00147     config_file_options.add(config).add(hidden);
00148
00149     boost::program_options::options_description visible ("Allowed options");
00150     visible.add(generic).add(config);
00151
00152     boost::program_options::positional_options_description p;

```

```

00146 p.add ("copyright", -1);
00147
00148 boost::program_options::variables_map vm;
00149 boost::program_options::
00150     store (boost::program_options::command_line_parser (argc, argv).
00151         options (cmdline_options).positional(p).run(), vm);
00152
00153 std::ifstream ifs ("airtsp.cfg");
00154 boost::program_options::store (parse_config_file (ifs, config_file_options),
00155     vm);
00156 boost::program_options::notify (vm);
00157
00158 if (vm.count ("help")) {
00159     std::cout << visible << std::endl;
00160     return K_AIRTSP_EARLY_RETURN_STATUS;
00161 }
00162
00163 if (vm.count ("version")) {
00164     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00165     return K_AIRTSP_EARLY_RETURN_STATUS;
00166 }
00167
00168 if (vm.count ("prefix")) {
00169     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00170     return K_AIRTSP_EARLY_RETURN_STATUS;
00171 }
00172
00173 if (vm.count ("builtin")) {
00174     ioIsBuiltin = true;
00175 }
00176 const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00177 std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00178
00179 //
00180 std::ostringstream oErrorMessageStr;
00181 oErrorMessageStr << "Either the -b/--builtin option, or the -i/--input option"
00182     << " must be specified";
00183
00184 if (ioIsBuiltin == false) {
00185     if (vm.count ("input")) {
00186         ioInputFilename = vm["input"].as< std::string >();
00187         std::cout << "Input filename is: " << ioInputFilename << std::endl;
00188     } else {
00189         // The built-in option is not selected. However, no schedule input file
00190         // is specified
00191         std::cerr << oErrorMessageStr.str() << std::endl;
00192     }
00193 }
00194
00195 //
00196 if (vm.count ("read_booking_request")) {
00197     ioReadBookingRequestFromCmdLine = true;
00198 }
00199
00200 const std::string readBookingRequestFromCmdLineStr =
00201     (ioReadBookingRequestFromCmdLine == true)?"yes":"no";
00202 std::cout << "A booking request is to be given as command-line option? "
00203     << readBookingRequestFromCmdLineStr << std::endl;
00204
00205 if (ioReadBookingRequestFromCmdLine == true) {
00206
00207     if (lWordList.empty() == true) {
00208         std::cerr << "When the --read_booking_request/-r option is given, "
00209             << "a query must also be provided (with the --bkg_req/-b "
00210             << "option at the end of the command-line)" << std::endl;
00211         return K_AIRTSP_EARLY_RETURN_STATUS;
00212     }

```

```

00213
00214     // Rebuild the booking request query string
00215     ioBookingRequestString = createStringFromWordList (lWordList);
00216     std::cout << "The booking request string is: " << ioBookingRequestString
00217         << std::endl;
00218 }
00219
00220 if (vm.count ("log")) {
00221     ioLogFilename = vm["log"].as< std::string >();
00222     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00223 }
00224
00225 return 0;
00226 }
00227
00228 // //////////////////////////////////////
00229 stdair::BookingRequestStruct
00230 parseBookingRequest (const std::string& iRequestOption) {
00231     typedef boost::tokenizer<boost::char_separator<char> > tokenizer;
00232     boost::char_separator<char> sep(" -");
00233
00234     tokenizer tokens (iRequestOption, sep);
00235
00236     // Origin (e.g., "NCE")
00237     tokenizer::iterator tok_iter = tokens.begin();
00238     assert (tok_iter != tokens.end());
00239     const stdair::AirportCode_T iOrigin (*tok_iter);
00240
00241     // Destination (e.g., "BKK")
00242     ++tok_iter; assert (tok_iter != tokens.end());
00243     const stdair::AirportCode_T iDestination (*tok_iter);
00244
00245     // POS (e.g., "NCE")
00246     ++tok_iter; assert (tok_iter != tokens.end());
00247     const stdair::AirportCode_T iPOS (*tok_iter);
00248
00249     // Preferred departure date (e.g., "2007-04-21")
00250     ++tok_iter; assert (tok_iter != tokens.end());
00251     const short lDepDateYear = boost::lexical_cast<short> (*tok_iter);
00252     ++tok_iter; assert (tok_iter != tokens.end());
00253     const short lDepDateMonth = boost::lexical_cast<short> (*tok_iter);
00254     ++tok_iter; assert (tok_iter != tokens.end());
00255     const short lDepDateDay = boost::lexical_cast<short> (*tok_iter);
00256     const stdair::Date_T iDepartureDate(lDepDateYear, lDepDateMonth, lDepDateDay);
00257
00258     // Request date (e.g., "2007-03-21")
00259     ++tok_iter; assert (tok_iter != tokens.end());
00260     const short lReqDateYear = boost::lexical_cast<short> (*tok_iter);
00261     ++tok_iter; assert (tok_iter != tokens.end());
00262     const short lReqDateMonth = boost::lexical_cast<short> (*tok_iter);
00263     ++tok_iter; assert (tok_iter != tokens.end());
00264     const short lReqDateDay = boost::lexical_cast<short> (*tok_iter);
00265     const stdair::Date_T iRequestDate (lReqDateYear, lReqDateMonth, lReqDateDay);
00266
00267     // Request time (e.g., "08:34:23")
00268     ++tok_iter; assert (tok_iter != tokens.end());
00269     const short lReqTimeHours = boost::lexical_cast<short> (*tok_iter);
00270     ++tok_iter; assert (tok_iter != tokens.end());
00271     const short lReqTimeMinutes = boost::lexical_cast<short> (*tok_iter);
00272     ++tok_iter; assert (tok_iter != tokens.end());
00273     const short lReqTimeSeconds = boost::lexical_cast<short> (*tok_iter);
00274     const stdair::Duration_T iRequestTime (lReqTimeHours, lReqTimeMinutes,
00275         lReqTimeSeconds);
00276
00277     // Request date-time (aggregation of the two items above)
00278     const stdair::DateTime_T iRequestDateTime (iRequestDate, iRequestTime);
00279

```

```

00280 // Preferred cabin (e.g., "C")
00281 ++tok_iter; assert (tok_iter != tokens.end());
00282 const stdair::CabinCode_T iPreferredCabin (*tok_iter);
00283
00284 // Party size (e.g., 1)
00285 ++tok_iter; assert (tok_iter != tokens.end());
00286 const stdair::NbOfSeats_T iPartySize = 1;
00287
00288 // Channel (e.g., "DF")
00289 ++tok_iter; assert (tok_iter != tokens.end());
00290 const stdair::ChannelLabel_T iChannel (*tok_iter);
00291
00292 // Trip type (e.g., "RO")
00293 ++tok_iter; assert (tok_iter != tokens.end());
00294 const stdair::TripType_T iTripType (*tok_iter);
00295
00296 // Stay duration (e.g., 5)
00297 ++tok_iter; assert (tok_iter != tokens.end());
00298 const stdair::DayDuration_T iStayDuration = 5;
00299
00300 // Frequent flyer (e.g., "NONE")
00301 ++tok_iter; assert (tok_iter != tokens.end());
00302 const stdair::FrequentFlyer_T iFrequentFlyerType ("NONE");
00303
00304 // Preferred departure time (e.g., "10:00:00")
00305 ++tok_iter; assert (tok_iter != tokens.end());
00306 const short lPrefTimeHours = boost::lexical_cast<short> (*tok_iter);
00307 ++tok_iter; assert (tok_iter != tokens.end());
00308 const short lPrefTimeMinutes = boost::lexical_cast<short> (*tok_iter);
00309 ++tok_iter; assert (tok_iter != tokens.end());
00310 const short lPrefTimeSeconds = boost::lexical_cast<short> (*tok_iter);
00311 const stdair::Duration_T iPreferredDepartureTime (lPrefTimeHours,
00312                                                    lPrefTimeMinutes,
00313                                                    lPrefTimeSeconds);
00314
00315 // Willingness-to-pay (e.g., 2000.0)
00316 ++tok_iter; assert (tok_iter != tokens.end());
00317 const stdair::WTP_T iWTP = 2000.0;
00318
00319 // Value of time (e.g., 20.0)
00320 ++tok_iter; assert (tok_iter != tokens.end());
00321 const stdair::PriceValue_T iValueOfTime = 20.0;
00322
00323 // Change fee acceptance (e.g., true)
00324 //++tok_iter; assert (tok_iter != tokens.end());
00325 const stdair::ChangeFees_T iChangeFees = true;
00326 const stdair::Disutility_T iChangeFeeDisutility = 50;
00327
00328 // Non refundable acceptance (e.g., true)
00329 //++tok_iter; assert (tok_iter != tokens.end());
00330 const stdair::NonRefundable_T iNonRefundable = true;
00331 const stdair::Disutility_T iNonRefundableDisutility = 50;
00332
00333 // Build and return the booking request structure
00334 return stdair::BookingRequestStruct (iOrigin,
00335                                     iDestination, iPOS,
00336                                     iDepartureDate, iRequestDateTime,
00337                                     iPreferredCabin, iPartySize,
00338                                     iChannel, iTripType, iStayDuration,
00339                                     iFrequentFlyerType,
00340                                     iPreferredDepartureTime, iWTP,
00341                                     iValueOfTime, iChangeFees,
00342                                     iChangeFeeDisutility, iNonRefundable,
00343                                     iNonRefundableDisutility);
00344 }
00345
00346 // ////////// M A I N //////////

```

```

00347 int main (int argc, char* argv[]) {
00348
00349     // State whether the BOM tree should be built-in or parsed from an
00350     // input file
00351     bool isBuiltin;
00352
00353     // A booking request should be given as command-line option
00354     bool readBookingRequestFromCmdLine;
00355
00356     // Input file name
00357     stdair::Filename_T lInputFilename;
00358
00359     // Output log File
00360     stdair::Filename_T lLogFilename;
00361
00362     // Booking request string
00363     std::string lBookingRequestString;
00364
00365     // Call the command-line option parser
00366     const int lOptionParserStatus =
00367         readConfiguration (argc, argv, isBuiltin, readBookingRequestFromCmdLine,
00368                             lInputFilename, lLogFilename, lBookingRequestString);
00369
00370     if (lOptionParserStatus == K_AIRTSP_EARLY_RETURN_STATUS) {
00371         return 0;
00372     }
00373
00374     // Set the log parameters
00375     std::ofstream logOutputFile;
00376     // Open and clean the log outputfile
00377     logOutputFile.open (lLogFilename.c_str());
00378     logOutputFile.clear();
00379
00380     // Initialise the AirTSP service object
00381     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00382     AIRTSP::AIRTSP_Service airtspService (lLogParams);
00383
00384     // Check wether or not (CSV) input files should be read
00385     if (isBuiltin == true) {
00386
00387         // Build the sample BOM tree
00388         airtspService.buildSampleBom();
00389
00390     } else {
00391         // Build the BOM tree from parsing input files
00392         const stdair::ScheduleFilePath lScheduleFilePath (lInputFilename);
00393         airtspService.parseAndLoad (lScheduleFilePath);
00394     }
00395
00396     // Check wether or not a booking request is given as a command-line option
00397     if (readBookingRequestFromCmdLine == false) {
00398         lBookingRequestString = K_AIRTSP_DEFAULT_BOOKING_REQUEST;
00399     }
00400
00401     // DEBUG
00402     STDAIR_LOG_DEBUG("Booking request string: '" << lBookingRequestString << "'");
00403
00404     // Create a booking request object
00405     const stdair::BookingRequestStruct& lBookingRequest =
00406         parseBookingRequest (lBookingRequestString);
00407
00408     //
00409     stdair::TravelSolutionList_T lTravelSolutionList;
00410     airtspService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
00411
00412     // DEBUG
00413     STDAIR_LOG_DEBUG ("Parsed booking request: " << lBookingRequest);

```



```
00414
00415 // DEBUG
00416 std::ostringstream oStream;
00417 stdair::BomDisplay::csvDisplay (oStream, lTravelSolutionList);
00418 STDAIR_LOG_DEBUG (oStream.str());
00419
00420 // Close the Log outputFile
00421 logOutputFile.close();
00422
00423 return 0;
00424 }
```

## 26.15 airtsp/batches/BookingRequestParser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <fstream>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#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>
#include <stdair/service/Logger.hpp>
#include <airtsp/batches/BookingRequestParser.hpp>
```

### Classes

- struct [airtsp::store\\_place\\_element](#)
- struct [airtsp::store\\_date](#)
- struct [airtsp::store\\_airline\\_sign](#)
- struct [airtsp::store\\_airline\\_code](#)
- struct [airtsp::store\\_airline\\_name](#)
- struct [airtsp::store\\_passenger\\_number](#)
- struct [airtsp::store\\_adult\\_passenger\\_type](#)
- struct [airtsp::store\\_child\\_passenger\\_type](#)
- struct [airtsp::store\\_pet\\_passenger\\_type](#)
- struct [airtsp::SearchStringParser](#)
- struct [airtsp::SearchStringParser::definition< ScannerT >](#)

### Namespaces

- namespace [airtsp](#)

### Defines

- `#define` [BOOST\\_SPIRIT\\_DEBUG](#)

## Typedefs

- typedef char [char\\_t](#)
- typedef char const \* [iterator\\_t](#)
- typedef boost::spirit::classic::scanner< [iterator\\_t](#) > [scanner\\_t](#)
- typedef boost::spirit::classic::rule< [scanner\\_t](#) > [rule\\_t](#)

## Functions

- [SearchString\\_T airtsp::parseBookingRequest](#) (const std::string &iSearchString)

## Variables

- boost::spirit::classic::int\_parser< unsigned int, 10, 1, 1 > [airtsp::int1\\_p](#)
- boost::spirit::classic::uint\_parser< unsigned int, 10, 1, 1 > [airtsp::uint1\\_p](#)
- boost::spirit::classic::uint\_parser< unsigned int, 10, 1, 2 > [airtsp::uint1\\_2\\_p](#)
- boost::spirit::classic::uint\_parser< int, 10, 2, 2 > [airtsp::uint2\\_p](#)
- boost::spirit::classic::uint\_parser< int, 10, 2, 4 > [airtsp::uint2\\_4\\_p](#)
- boost::spirit::classic::uint\_parser< int, 10, 4, 4 > [airtsp::uint4\\_p](#)
- boost::spirit::classic::uint\_parser< int, 10, 1, 4 > [airtsp::uint1\\_4\\_p](#)

### 26.15.1 Define Documentation

#### 26.15.1.1 #define BOOST\_SPIRIT\_DEBUG

Definition at line 12 of file [BookingRequestParser.cpp](#).

### 26.15.2 Typedef Documentation

#### 26.15.2.1 typedef char char\_t

Definition at line 28 of file [BookingRequestParser.cpp](#).

#### 26.15.2.2 typedef char const\* iterator\_t

Definition at line 29 of file [BookingRequestParser.cpp](#).

#### 26.15.2.3 typedef boost::spirit::classic::scanner<iterator\_t> scanner\_t

Definition at line 31 of file [BookingRequestParser.cpp](#).

#### 26.15.2.4 typedef boost::spirit::classic::rule<scanner\_t> rule\_t

Definition at line 32 of file [BookingRequestParser.cpp](#).

## 26.16 BookingRequestParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <fstream>
00008 // Boost (Extended STL)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 // Boost Spirit (Parsing)
00012 #define BOOST_SPIRIT_DEBUG
00013 #include <boost/spirit/home/classic/core.hpp>
00014 #include <boost/spirit/home/classic/attribute.hpp>
00015 #include <boost/spirit/home/classic/utility/functor_parser.hpp>
00016 #include <boost/spirit/home/classic/utility/loops.hpp>
00017 #include <boost/spirit/home/classic/utility/chset.hpp>
00018 #include <boost/spirit/home/classic/utility/confix.hpp>
00019 #include <boost/spirit/home/classic/iterator/file_iterator.hpp>
00020 #include <boost/spirit/home/classic/actor/push_back_actor.hpp>
00021 #include <boost/spirit/home/classic/actor/assign_actor.hpp>
00022 // StdAir
00023 #include <stdair/service/Logger.hpp>
00024 // AirTSP
00025 #include <airtsp/batches/BookingRequestParser.hpp>
00026
00027 // Type definitions
00028 typedef char char_t;
00029 typedef char const* iterator_t;
00030 //typedef boost::spirit::classic::file_iterator<char_t> iterator_t;
00031 typedef boost::spirit::classic::scanner<iterator_t> scanner_t;
00032 typedef boost::spirit::classic::rule<scanner_t> rule_t;
00033
00034 namespace airtsp {
00035
00036     struct store_place_element {
00037         store_place_element (SearchString_T& ioSearchString)
00038             : _searchString (ioSearchString) {}
00039
00040         void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00041             std::string lPlace (iStr, iStrEnd);
00042             // std::cout << "Place: " << lPlace << std::endl;
00043
00044             // Set the place
00045             _searchString._tmpPlace._name += " " + lPlace;
00046
00047             // Add the parsed place to the list
00048             // _searchString._placeList.push_back (_searchString._tmpPlace);
00049         }
00050
00051         SearchString_T& _searchString;
00052     };
00053
00054     struct store_date {
00055         store_date (SearchString_T& ioSearchString)
00056             : _searchString (ioSearchString) {}
00057
00058         void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00059             _searchString._tmpDate._date = _searchString._tmpDate.getDate();
00060             // std::cout << "Board date: "
00061             // << _searchString._date << std::endl;
00062
00063             // Add the parsed date to the list
00064             _searchString._dateList.push_back (_searchString._tmpDate);
00065         }
00066     };

```

```

00072
00073     SearchString_T& _searchString;
00074 };
00075
00076 struct store_airline_sign {
00077     store_airline_sign (SearchString_T& ioSearchString)
00078         : _searchString (ioSearchString) {}
00079
00080     void operator() (bool iAirlineSign) const {
00081         _searchString._tmpAirline._isPreferred = !iAirlineSign;
00082         // std::cout << "Airline is preferred: " << iAirlineSign << std::endl;
00083     }
00084
00085     SearchString_T& _searchString;
00086 };
00087
00088 struct store_airline_code {
00089     store_airline_code (SearchString_T& ioSearchString)
00090         : _searchString (ioSearchString) {}
00091
00092     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00093         std::string lAirlineCode (iStr, iStrEnd);
00094         _searchString._tmpAirline._code = lAirlineCode;
00095         // std::cout << "Airline code: " << lAirlineCode << std::endl;
00096
00097         // Add the parsed airline to the list
00098         _searchString._airlineList.push_back (_searchString._tmpAirline);
00099     }
00100
00101     SearchString_T& _searchString;
00102 };
00103
00104 struct store_airline_name {
00105     store_airline_name (SearchString_T& ioSearchString)
00106         : _searchString (ioSearchString) {}
00107
00108     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00109         std::string lAirlineName (iStr, iStrEnd);
00110         _searchString._tmpAirline._name = lAirlineName;
00111         // std::cout << "Airline: " << lAirlineName << std::endl;
00112
00113         // Add the parsed airline to the list
00114         _searchString._airlineList.push_back (_searchString._tmpAirline);
00115     }
00116
00117     SearchString_T& _searchString;
00118 };
00119
00120 struct store_passenger_number {
00121     store_passenger_number (SearchString_T& ioSearchString)
00122         : _searchString (ioSearchString) {}
00123
00124     void operator() (unsigned int iNumber) const {
00125         _searchString._tmpPassenger._number = iNumber;
00126         // std::cout << "Number of passengers: " << iNumber << std::endl;
00127     }
00128
00129     SearchString_T& _searchString;
00130 };
00131
00132 struct store_adult_passenger_type {
00133     store_adult_passenger_type (SearchString_T& ioSearchString)
00134         : _searchString (ioSearchString) {}
00135
00136     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00137         std::string lPassengerType (iStr, iStrEnd);
00138         _searchString._tmpPassenger._type = Passenger_T::ADULT;

```

```

00154         // std::cout << "Passenger type: " << lPassengerType << std::endl;
00155
00156         // Add the parsed passenger to the list
00157         _searchString._passengerList.push_back (_searchString._tmpPassenger);
00158     }
00159
00160     SearchString_T& _searchString;
00161 };
00162
00163 struct store_child_passenger_type {
00164     store_child_passenger_type (SearchString_T& ioSearchString)
00165         : _searchString (ioSearchString) {}
00166
00167     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00168         std::string lPassengerType (iStr, iStrEnd);
00169         _searchString._tmpPassenger._type = Passenger_T::CHILD;
00170         // std::cout << "Passenger type: " << lPassengerType << std::endl;
00171
00172         // Add the parsed passenger to the list
00173         _searchString._passengerList.push_back (_searchString._tmpPassenger);
00174     }
00175
00176     SearchString_T& _searchString;
00177 };
00178
00179 struct store_pet_passenger_type {
00180     store_pet_passenger_type (SearchString_T& ioSearchString)
00181         : _searchString (ioSearchString) {}
00182
00183     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00184         std::string lPassengerType (iStr, iStrEnd);
00185         _searchString._tmpPassenger._type = Passenger_T::PET;
00186         // std::cout << "Passenger type: " << lPassengerType << std::endl;
00187
00188         // Add the parsed passenger to the list
00189         _searchString._passengerList.push_back (_searchString._tmpPassenger);
00190     }
00191
00192     SearchString_T& _searchString;
00193 };
00194
00195 // ////////////////////////////////// Utilities //////////////////////////////////
00196 boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p;
00197 boost::spirit::classic::uint_parser<unsigned int, 10, 1, 1> uint1_p;
00198 boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> uint1_2_p;
00199 boost::spirit::classic::uint_parser<int, 10, 2, 2> uint2_p;
00200 boost::spirit::classic::uint_parser<int, 10, 2, 4> uint2_4_p;
00201 boost::spirit::classic::uint_parser<int, 10, 4, 4> uint4_p;
00202 boost::spirit::classic::uint_parser<int, 10, 1, 4> uint1_4_p;
00203
00204 //
00205 // Our calculator grammar (using subrules)
00206 //
00207
00208 using namespace boost::spirit::classic;
00209
00210 struct SearchStringParser :
00211     public boost::spirit::classic::grammar<SearchStringParser> {
00212     SearchStringParser (SearchString_T& ioSearchString)
00213         : _searchString (ioSearchString) {}
00214
00215     template <typename ScannerT>
00216     struct definition {
00217         definition (SearchStringParser const& self) {

```

```

00262         search_string = places
00263         >> !( dates )
00264         >> *( preferred_airlines )
00265         >> *( passengers )
00266         ;
00267
00268         places =
00269         +( place_element )
00270         ;
00271
00272         place_element =
00273         lexeme_d[ (repeat_p(1,20) [chset_p("a-z")]) [store_place_element(self.
00274         _searchString)] ]
00275         ;
00276
00277         dates =
00278         date[store_date(self._searchString)]
00279         >> !date[store_date(self._searchString)]
00280         ;
00281
00282         date =
00283         ( month | day )
00284         >> boost::spirit::classic::chset_p("/")
00285         >> ( day | month )
00286         >> ! ( boost::spirit::classic::chset_p("/")
00287         >> year )
00288         ;
00289
00290         day =
00291         lexeme_d[ limit_d(1u,31u) [uint1_2_p] [assign_a(self._searchString.
00292         _tmpDate._day)] ]
00293         ;
00294
00295         month =
00296         lexeme_d[ limit_d(1u,12u) [uint1_2_p] [assign_a(self._searchString.
00297         _tmpDate._month)] ]
00298         ;
00299
00300         year =
00301         lexeme_d[ limit_d(2000u,2099u) [uint4_p] [assign_a(self._searchString.
00302         _tmpDate._year)] ]
00303         | lexeme_d[ limit_d(0u,99u) [uint2_p] [assign_a(self._searchString.
00304         _tmpDate._year)] ]
00305         ;
00306
00307         preferred_airlines =
00308         !(boost::spirit::classic::sign_p) [store_airline_sign(self.
00309         _searchString)]
00310         >> airline_code | airline_name
00311         ;
00312
00313         airline_code =
00314         lexeme_d[ (repeat_p(2,3) [chset_p("0-9a-z")]) [store_airline_code(self.
00315         _searchString)] ]
00316         ;
00317
00318         airline_name =
00319         lexeme_d[ (repeat_p(4,20) [chset_p("0-9a-z")]) [store_airline_name(self.
00320         _searchString)] ]
00321         ;
00322
00323         passengers =
00324         passenger_number >> passenger_type
00325         ;
00326
00327         passenger_number =
00328         lexeme_d[ limit_d(1u, 9u) [uint1_p] [store_passenger_number(self.

```



```

    _searchString)] ]
00321         ;
00322
00323     passenger_type =
00324         passenger_adult_type[store_adult_passenger_type(self._searchString)]
00325         | passenger_child_type[store_child_passenger_type(self._searchString)]
00326         | passenger_pet_type[store_pet_passenger_type(self._searchString)]
00327         ;
00328
00329     passenger_adult_type =
00330         lexeme_d[ as_lower_d [ str_p("adult") >> !ch_p('s') ] ]
00331         ;
00332
00333     passenger_child_type =
00334         lexeme_d[ as_lower_d [ str_p("child") >> !str_p("ren") ] ]
00335         ;
00336
00337     passenger_pet_type =
00338         lexeme_d[ as_lower_d [ str_p("dog") | str_p("cat") >> !ch_p('s') ] ]
00339         ;
00340
00341     BOOST_SPIRIT_DEBUG_NODE (search_string);
00342     BOOST_SPIRIT_DEBUG_NODE (places);
00343     BOOST_SPIRIT_DEBUG_NODE (place_element);
00344     BOOST_SPIRIT_DEBUG_NODE (dates);
00345     BOOST_SPIRIT_DEBUG_NODE (date);
00346     BOOST_SPIRIT_DEBUG_NODE (day);
00347     BOOST_SPIRIT_DEBUG_NODE (month);
00348     BOOST_SPIRIT_DEBUG_NODE (year);
00349     BOOST_SPIRIT_DEBUG_NODE (preferred_airlines);
00350     BOOST_SPIRIT_DEBUG_NODE (airline_code);
00351     BOOST_SPIRIT_DEBUG_NODE (airline_name);
00352     BOOST_SPIRIT_DEBUG_NODE (passengers);
00353     BOOST_SPIRIT_DEBUG_NODE (passenger_number);
00354     BOOST_SPIRIT_DEBUG_NODE (passenger_type);
00355     BOOST_SPIRIT_DEBUG_NODE (passenger_adult_type);
00356     BOOST_SPIRIT_DEBUG_NODE (passenger_child_type);
00357     BOOST_SPIRIT_DEBUG_NODE (passenger_pet_type);
00358 }
00359
00360     boost::spirit::classic::rule<ScannerT> search_string, places,
place_element,
00361     dates, date, month, day, year,
00362     preferred_airlines, airline_code, airline_name,
00363     passengers, passenger_number, passenger_type, passenger_adult_type,
00364     passenger_child_type, passenger_pet_type;
00365
00366     boost::spirit::classic::rule<ScannerT> const& start() const { return
search_string; }
00367 };
00368
00369     SearchString_T& _searchString;
00370 };
00371
00372 // //////////////////////////////////////
00373 SearchString_T parseBookingRequest (const std::string& iSearchString) {
00374     SearchString_T oSearchStringStruct;
00375
00376     // Read the search string
00377     iterator_t lStringIterator = iSearchString.c_str();
00378
00379     // Instantiate the structure that will hold the result of the parsing.
00380     SearchStringParser lSearchStringParser (oSearchStringStruct);
00381     boost::spirit::classic::parse_info<iterator_t> info =
00382         boost::spirit::classic::parse (lStringIterator, lSearchStringParser,
00383                                         boost::spirit::classic::space_p);
00384

```

```
00385     STDAIR_LOG_DEBUG ("-----");
00386
00387     bool hasBeenParsingSuccessful = info.full;
00388     if (hasBeenParsingSuccessful == true) {
00389         STDAIR_LOG_DEBUG ("Parsing succeeded");
00390
00391     } else {
00392         STDAIR_LOG_DEBUG ("Parsing failed");
00393     }
00394     STDAIR_LOG_DEBUG ("-----");
00395
00396     return oSearchStringStruct;
00397 }
00398
00399 }
```

## 26.17 airtsp/batches/BookingRequestParser.hpp File Reference

```
#include <string>
#include <vector>
```

### Classes

- struct [airtsp::Place\\_T](#)
- struct [airtsp::Date\\_T](#)
- struct [airtsp::Airline\\_T](#)
- struct [airtsp::Passenger\\_T](#)
- struct [airtsp::SearchString\\_T](#)

### Namespaces

- namespace [airtsp](#)

### Typedefs

- typedef std::vector< Place\_T > [airtsp::PlaceList\\_T](#)
- typedef std::vector< Date\_T > [airtsp::DateList\\_T](#)
- typedef std::vector< Airline\_T > [airtsp::AirlineList\\_T](#)
- typedef std::vector< Passenger\_T > [airtsp::PassengerList\\_T](#)

### Functions

- SearchString\_T [airtsp::parseBookingRequest](#) (const std::string &iSearchString)

## 26.18 BookingRequestParser.hpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <string>
00006 #include <vector>
00007
00008 namespace airtsp {
00009
00011     struct Place_T {
00012         // Attributes
00013         std::string _name;
00014         std::string _code;
00016         Place_T () : _name (""), _code ("") {}
00017         /* Display. */
00018         void display() const {
00019             std::cout << "Place: " << _name << " (" << _code << ")" << std::endl;
00020         }
00021     };
00022
00024     typedef std::vector<Place_T> PlaceList_T;
00025
00027     struct Date_T {
00028         // Attributes
00029         boost::gregorian::date _date;
00030         unsigned int _reldays;
00031         unsigned int _day;
00032         unsigned int _month;
00033         unsigned int _year;
00035         Date_T () : _reldays (14), _day(1), _month(1), _year(1970) {}
00036         /* Display. */
00037         void display() const {
00038             std::cout << "Date: " << _date << " (" << _day << "/" << _month
00039                 << "/" << _year << "), i.e. in " << _reldays << " days"
00040                 << std::endl;
00041         }
00043         boost::gregorian::date getDate() const {
00044             return boost::gregorian::date (_year, _month, _day);
00045         }
00046     };
00047
00049     typedef std::vector<Date_T> DateList_T;
00050
00052     struct Airline_T {
00053         // Attributes
00054         bool _isPreferred;
00055         std::string _name;
00056         std::string _code;
00058         Airline_T () : _isPreferred (true), _name(""), _code("") {}
00059         /* Display. */
00060         void display() const {
00061             const std::string isPreferredStr = (_isPreferred)?"+":"-";
00062             std::cout << "Airline: " << isPreferredStr << _name << " (" << _code << ")"
00063
00064                 << std::endl;
00065         }
00066     };
00068     typedef std::vector<Airline_T> AirlineList_T;
00069
00071     struct Passenger_T {
00072         // Attributes
00073         typedef enum { ADULT = 0, CHILD, PET, LAST_VALUE } PassengerType_T;
00074         static const std::string _labels[LAST_VALUE];
00075         PassengerType_T _type;

```

```

00076     unsigned short _number;
00077     Passenger_T () : _type(ADULT), _number(1) {}
00078     /* Display. */
00079     void display() const {
00080         std::cout << "Passenger: " << _number << " (" << _labels[_type] << ")"
00081             << std::endl;
00082     }
00083 };
00084 };
00085
00086 const std::string Passenger_T::_labels[Passenger_T::LAST_VALUE] =
00087     { "Adult", "Child", "Pet" };
00088
00089 typedef std::vector<Passenger_T> PassengerList_T;
00090
00091 struct SearchString_T {
00092     // Attributes
00093     PlaceList_T _placeList;
00094     DateList_T _dateList;
00095     AirlineList_T _airlineList;
00096     PassengerList_T _passengerList;
00097
00098     SearchString_T () {}
00099
00100     /* Display. */
00101     void display() const {
00102         std::cout << std::endl;
00103
00104         for (PlaceList_T::const_iterator itPlace = _placeList.begin();
00105             itPlace != _placeList.end(); ++itPlace) {
00106             const Place_T& lPlace = *itPlace;
00107             lPlace.display();
00108         }
00109
00110         for (DateList_T::const_iterator itDate = _dateList.begin();
00111             itDate != _dateList.end(); ++itDate) {
00112             const Date_T& lDate = *itDate;
00113             lDate.display();
00114         }
00115
00116         for (AirlineList_T::const_iterator itAirline = _airlineList.begin();
00117             itAirline != _airlineList.end(); ++itAirline) {
00118             const Airline_T& lAirline = *itAirline;
00119             lAirline.display();
00120         }
00121
00122         for (PassengerList_T::const_iterator itPassenger = _passengerList.begin();
00123             itPassenger != _passengerList.end(); ++itPassenger) {
00124             const Passenger_T& lPassenger = *itPassenger;
00125             lPassenger.display();
00126         }
00127
00128         std::cout << "-- Staging --" << std::endl;
00129         _tmpPlace.display();
00130     }
00131
00132     // //// Staging ////
00133     Place_T _tmpPlace;
00134     Date_T _tmpDate;
00135     Airline_T _tmpAirline;
00136     Passenger_T _tmpPassenger;
00137 };
00138
00139 //
00140 // The booking request grammar (using subrules)
00141 //
00142
00143 SearchString_T parseBookingRequest (const std::string& iSearchString);

```

```
00175  
00176 }
```

## 26.19 airtsp/bom/AirportList.hpp File Reference

```
#include <set>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef std::set< stdair::AirportCode\_T > [AIRTSP::AirportList\\_T](#)
- typedef std::vector< stdair::AirportCode\_T > [AIRTSP::AirportOrderedList\\_T](#)

## 26.20 AirportList.hpp

```
00001 #ifndef __AIRTSP_BOM_AIRPORTLIST_HPP
00002 #define __AIRTSP_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 AIRTSP {
00014
00016     typedef std::set<stdair::AirportCode_T> AirportList_T;
00017     typedef std::vector<stdair::AirportCode_T> AirportOrderedList_T;
00018
00019 }
00020 #endif // __AIRTSP_BOM_AIRPORTLIST_HPP
```



## 26.21 airtsp/bom/BomDisplay.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <airtsp/bom/ReachableUniverse.hpp>
#include <airtsp/bom/BomDisplay.hpp>
```

### Classes

- struct [AIRTSP::FlagSaver](#)

### Namespaces

- namespace [AIRTSP](#)

## 26.22 BomDisplay.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 // AirTSP
00012 #include <airtsp/bom/ReachableUniverse.hpp>
00013 #include <airtsp/bom/BomDisplay.hpp>
00014
00015 namespace AIRTSP {
00016
00022     struct FlagSaver {
00023     public:
00025         FlagSaver (std::ostream& oStream)
00026             : _oStream (oStream), _streamFlags (oStream.flags()) {
00027         }
00028
00030         ~FlagSaver() {
00031             // Reset formatting flags of the given output stream
00032             _oStream.flags (_streamFlags);
00033         }
00034
00035     private:
00037         std::ostream& _oStream;
00039         std::ios::fmtflags _streamFlags;
00040     };
00041
00042 // //////////////////////////////////////
00043 std::string BomDisplay::csvDisplay (const stdair::BomRoot& iBomRoot) {
00044     std::ostringstream oStream;
00045
00049     oStream << std::endl;
00050     oStream << "=====
00051         << std::endl;
00052     oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
00053     oStream << "=====
00054         << std::endl;
00055
00056     // Check whether there are ReachableUniverse objects
00057     if (stdair::BomManager::hasList<ReachableUniverse> (iBomRoot) == false) {
00058         return oStream.str();
00059     }
00060
00061     // Retrieve the ReachableUniverse list
00062     const ReachableUniverseList_T& lReachableUniverseList =
00063         stdair::BomManager::getList<ReachableUniverse> (iBomRoot);
00064
00065     // Browse the networks for each departure airport
00066     for (ReachableUniverseList_T::const_iterator itReachableUniverse =
00067         lReachableUniverseList.begin();
00068         itReachableUniverse != lReachableUniverseList.end();
00069         ++itReachableUniverse) {
00070         ReachableUniverse* lReachableUniverse_ptr = *itReachableUniverse;
00071         assert (lReachableUniverse_ptr != NULL);
00072
00073         // Display the reachable universe
00074         csvDisplay (oStream, *lReachableUniverse_ptr);
00075     }
00076
00077     return oStream.str();

```

```
00078     }
00079
00080     // //////////////////////////////////////
00081     void BomDisplay::csvDisplay (std::ostream& oStream,
00082                                 const ReachableUniverse& iReachableUniverse) {
00083         // Save the formatting flags for the given STL output stream
00084         FlagSaver flagSaver (oStream);
00085
00086         oStream << "+++++" << std::endl;
00087         oStream << iReachableUniverse.toString();
00088         oStream << "+++++" << std::endl;
00089     }
00090
00091 }
```

## 26.23 airtsp/bom/BomDisplay.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
```

### Classes

- class [AIRTSP::BomDisplay](#)  
*Utility class to display AirTSP objects with a pretty format.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.24 BomDisplay.hpp

```
00001 #ifndef __AIRTSP_BOM_BOMDISPLAY_HPP
00002 #define __AIRTSP_BOM_BOMDISPLAY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Airtsp
00011
00012 namespace stdair {
00013     class BomRoot;
00014 }
00015
00016 namespace AIRTSP {
00017     class ReachableUniverse;
00018
00019     class BomDisplay {
00020     public:
00021         // ////////////////////////////////// Display support methods //////////////////////////////////
00022         static std::string csvDisplay (const stdair::BomRoot&);
00023
00024         static void csvDisplay (std::ostream&, const ReachableUniverse&);
00025     };
00026 }
00027
00028 #endif // __AIRTSP_BOM_BOMDISPLAY_HPP
```

## 26.25 airtsp/bom/FareFamilyStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <airtsp/bom/FareFamilyStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.26 FareFamilyStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // AirTSP
00008 #include <airtsp/bom/FareFamilyStruct.hpp>
00009
00010 namespace AIRTSP {
00011
00012 // //////////////////////////////////////
00013 FareFamilyStruct::
00014 FareFamilyStruct (const stdair::FamilyCode_T& iFamilyCode,
00015                  const stdair::CurveKey_T& iFRAT5Key,
00016                  const stdair::CurveKey_T& iFFDisutilityKey,
00017                  const stdair::ClassList_String_T& iClasses)
00018 : _familyCode (iFamilyCode), _frat5CurveKey (iFRAT5Key),
00019   _ffDisutilityCurveKey (iFFDisutilityKey), _classes (iClasses) {
00020 }
00021
00022 // //////////////////////////////////////
00023 const std::string FareFamilyStruct::describe() const {
00024     std::ostringstream ostr;
00025     ostr << "          " << _familyCode << " "
00026           << _frat5CurveKey << " " << _ffDisutilityCurveKey
00027           << " " << _classes << ", ";
00028     return ostr.str();
00029 }
00030
00031 }

```

## 26.27 airtsp/bom/FareFamilyStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

### Classes

- struct [AIRTSP::FareFamilyStruct](#)

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef std::vector< FareFamilyStruct > [AIRTSP::FareFamilyStructList\\_T](#)



## 26.28 FareFamilyStruct.hpp

```

00001 #ifndef __AIRTSP_BOM_FAREFAMILYSTRUCT_HPP
00002 #define __AIRTSP_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
00014 namespace AIRTSP {
00015
00016     struct FareFamilyStruct : public stdair::StructAbstract {
00017         // Attributes
00018         stdair::FamilyCode_T _familyCode;
00019         stdair::CurveKey_T _frat5CurveKey;
00020         stdair::CurveKey_T _ffDisutilityCurveKey;
00021         stdair::ClassList_String_T _classes;
00022
00023         FareFamilyStruct (const stdair::FamilyCode_T&,
00024                          const stdair::CurveKey_T&, const stdair::CurveKey_T&,
00025                          const stdair::ClassList_String_T&);
00026
00027         const std::string describe() const;
00028     };
00029
00030     typedef std::vector<FareFamilyStruct> FareFamilyStructList_T;
00031
00032 }
00033 #endif // __AIRTSP_BOM_FAREFAMILYSTRUCT_HPP

```

## 26.29 airtsp/bom/FlightPeriodStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/AIRTSP_Types.hpp>
#include <airtsp/bom/FlightPeriodStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.30 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 // AirTSP
00011 #include <airtsp/AIRTSP_Types.hpp>
00012 #include <airtsp/bom/FlightPeriodStruct.hpp>
00013
00014 namespace AIRTSP {
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 (itSegment == _segmentList.end()) {
00128         std::ostringstream ostr;
00129         ostr << "Within the schedule input file, there is a flight, for which "
00130             << "the airports of segments and those of the legs "
00131             << "do not correspond";
00132         STDAIR_LOG_ERROR (ostr.str());
00133     }

```

```

00138         throw SegmentDateNotFoundException (oStr.str());
00139     }
00140
00141     // Add the Cabin structure to the Segment Cabin structure.
00142     assert (itSegment != _segmentList.end());
00143     SegmentStruct& lSegment = *itSegment;
00144     lSegment._cabinList.push_back (iCabin);
00145 }
00146
00147 // //////////////////////////////////////
00148 void FlightPeriodStruct::
00149 addSegmentCabin (const SegmentCabinStruct& iCabin) {
00150     // Iterate on all the Segment structures (as they get the same cabin
00151     // definitions)
00152     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00153          itSegment != _segmentList.end(); ++itSegment) {
00154         SegmentStruct& lSegment = *itSegment;
00155
00156         lSegment._cabinList.push_back (iCabin);
00157     }
00158 }
00159
00160 // //////////////////////////////////////
00161 void FlightPeriodStruct::
00162 addFareFamily (const SegmentStruct& iSegment,
00163               const SegmentCabinStruct& iCabin,
00164               const FareFamilyStruct& iFareFamily) {
00165     // Retrieve the Segment structure corresponding to the (boarding, off) point
00166     // pair.
00167     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00168     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00169         const SegmentStruct& lSegment = *itSegment;
00170
00171         const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00172         const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00173         if (lSegment._boardingPoint == lBoardingPoint
00174             && lSegment._offPoint == lOffPoint) {
00175             break;
00176         }
00177     }
00178
00179     if (itSegment == _segmentList.end()) {
00180         std::ostringstream oStr;
00181         oStr << "Within the schedule input file, there is a flight, for which "
00182             << "the airports of segments and those of the legs "
00183             << "do not correspond";
00184         STDAIR_LOG_ERROR (oStr.str());
00185         throw SegmentDateNotFoundException (oStr.str());
00186     }
00187
00188     // Add the Cabin structure to the Segment Cabin structure.
00189     assert (itSegment != _segmentList.end());
00190     SegmentStruct& lSegment = *itSegment;
00191
00192     // Retrieve the Segment cabin structure given the cabin code
00193     SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00194     for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00195         const SegmentCabinStruct& lCabin = *itCabin;
00196
00197         const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00198         if (iCabin._cabinCode == lCabinCode) {
00199             break;
00200         }
00201     }
00202
00203     if (itCabin == lSegment._cabinList.end()) {
00204         std::ostringstream oStr;

```

```

00215         ostr << "Within the schedule input file, there is a flight "
00216         << "for which the cabin code does not exist.";
00217         STDAIR_LOG_ERROR (ostr.str());
00218         throw SegmentDateNotFoundException (ostr.str());
00219     }
00220
00221     // Add the Cabin structure to the Segment Cabin structure.
00222     assert (itCabin != lSegment._cabinList.end());
00223     SegmentCabinStruct& lCabin = *itCabin;
00224     lCabin._fareFamilies.push_back(iFareFamily);
00225 }
00226
00227 // //////////////////////////////////////
00228 void FlightPeriodStruct::
00229 addFareFamily (const SegmentCabinStruct& iCabin,
00230               const FareFamilyStruct& iFareFamily) {
00231     // Iterate on all the Segment structures (as they get the same cabin
00232     // definitions)
00233
00234     for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
00235          itSegment != _segmentList.end(); ++itSegment) {
00236         SegmentStruct& lSegment = *itSegment;
00237
00238         // Retrieve the Segment cabin structure given the cabin code
00239         SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
00240         for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00241             const SegmentCabinStruct& lCabin = *itCabin;
00242
00243             const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00244             if (iCabin._cabinCode == lCabinCode) {
00245                 break;
00246             }
00247         }
00248
00249         if (itCabin == lSegment._cabinList.end()) {
00250             std::ostringstream ostr;
00251             ostr << "Within the schedule input file, there is a flight "
00252             << "for which the cabin code does not exist.";
00253             STDAIR_LOG_ERROR (ostr.str());
00254             throw SegmentDateNotFoundException (ostr.str());
00255         }
00256
00257         // Add the Cabin structure to the Segment Cabin structure.
00258         assert (itCabin != lSegment._cabinList.end());
00259         SegmentCabinStruct& lCabin = *itCabin;
00260         lCabin._fareFamilies.push_back(iFareFamily);
00261     }
00262 }
00263
00264 }
00265
00266 }
00267
00268 }
00269 }

```

## 26.31 airtsp/bom/FlightPeriodStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airtsp/bom/LegCabinStruct.hpp>
#include <airtsp/bom/LegStruct.hpp>
#include <airtsp/bom/SegmentStruct.hpp>
#include <airtsp/bom/SegmentCabinStruct.hpp>
#include <airtsp/bom/FareFamilyStruct.hpp>
#include <airtsp/bom/AirportList.hpp>
```

### Classes

- struct [AIRTSP::FlightPeriodStruct](#)

### Namespaces

- namespace [AIRTSP](#)

## 26.32 FlightPeriodStruct.hpp

```

00001 #ifndef __AIRTSP_BOM_FLIGHTPERIODSTRUCT_HPP
00002 #define __AIRTSP_BOM_FLIGHTPERIODSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/basic/StructAbstract.hpp>
00012 #include <stdair/bom/DoWStruct.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/LegCabinStruct.hpp>
00015 #include <airtsp/bom/LegStruct.hpp>
00016 #include <airtsp/bom/SegmentStruct.hpp>
00017 #include <airtsp/bom/SegmentCabinStruct.hpp>
00018 #include <airtsp/bom/FareFamilyStruct.hpp>
00019 #include <airtsp/bom/AirportList.hpp>
00020
00021 namespace AIRTSP {
00022
00026     struct FlightPeriodStruct : public stdair::StructAbstract {
00027
00029         stdair::Date_T getDate() const;
00030
00032         stdair::Duration_T getTime() const;
00033
00035         const std::string describe() const;
00036
00039         void addAirport (const stdair::AirportCode_T&);
00040
00042         void buildSegments();
00043
00050         void addSegmentCabin (const SegmentStruct&,
00051                             const SegmentCabinStruct&);
00052
00058         void addSegmentCabin (const SegmentCabinStruct&);
00059
00066         void addFareFamily (const SegmentStruct&,
00067                             const SegmentCabinStruct&,
00068                             const FareFamilyStruct&);
00069
00075         void addFareFamily (const SegmentCabinStruct&,
00076                             const FareFamilyStruct&);
00077
00081         FlightPeriodStruct();
00082
00083         // Attributes
00084         stdair::AirlineCode_T _airlineCode;
00085         stdair::FlightNumber_T _flightNumber;
00086         stdair::DatePeriod_T _dateRange;
00087         stdair::DoWStruct _dow;
00088         LegStructList_T _legList;
00089         SegmentStructList_T _segmentList;
00090
00093         bool _legAlreadyDefined;
00094         LegStruct _itLeg;
00095         LegCabinStruct _itLegCabin;
00096
00098         stdair::Date_T _dateRangeStart;
00099         stdair::Date_T _dateRangeEnd;
00100         unsigned int _itYear;
00101         unsigned int _itMonth;
00102         unsigned int _itDay;

```



```
00103     int _dateOffset;
00104
00106     long _itHours;
00107     long _itMinutes;
00108     long _itSeconds;
00109
00112     AirportList_T _airportList;
00113     AirportOrderedList_T _airportOrderedList;
00114
00116     bool _areSegmentDefinitionsSpecific;
00117     SegmentStruct _itSegment;
00118     SegmentCabinStruct _itSegmentCabin;
00119 };
00120
00121 }
00122 #endif // __AIRTSP_BOM_FLIGHTPERIODSTRUCT_HPP
```

## 26.33 airtsp/bom/LegCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/LegCabin.hpp>
#include <airtsp/bom/LegCabinStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.34 LegCabinStruct.cpp

```
00001 // ////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/LegCabin.hpp>
00009 // AirTSP
00010 #include <airtsp/bom/LegCabinStruct.hpp>
00011
00012 namespace AIRTSP {
00013
00014 // ////////////////////////////////////////
00015 const std::string LegCabinStruct::describe() const {
00016     std::ostringstream ostr;
00017     ostr << "          " << _cabinCode << " " << _capacity << ", ";
00018     return ostr.str();
00019 }
00020
00021 // ////////////////////////////////////////
00022 void LegCabinStruct::fill (stdair::LegCabin& ioLegCabin) const {
00023     // Set the Capacity
00024     ioLegCabin.setCapacities (_capacity);
00025 }
00026
00027 }
```

## 26.35 airtsp/bom/LegCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

### Classes

- struct [AIRTSP::LegCabinStruct](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

### Typedefs

- typedef std::vector< LegCabinStruct > [AIRTSP::LegCabinStructList\\_T](#)

## 26.36 LegCabinStruct.hpp

```
00001 #ifndef __AIRTSP_BOM_LEGCABINSTRUCT_HPP
00002 #define __AIRTSP_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
00014 // Forward declarations
00015 namespace stdair {
00016     class LegCabin;
00017 }
00018
00019 namespace AIRTSP {
00020
00022     struct LegCabinStruct : public stdair::StructAbstract {
00023         // Attributes
00024         stdair::CabinCode_T _cabinCode;
00025         stdair::CabinCapacity_T _capacity;
00026
00029         void fill (stdair::LegCabin&) const;
00030
00032         const std::string describe() const;
00033     };
00034
00036     typedef std::vector<LegCabinStruct> LegCabinStructList_T;
00037
00038 }
00039 #endif // __AIRTSP_BOM_LEGCABINSTRUCT_HPP
```

## 26.37 airtsp/bom/LegStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airtsp/bom/LegStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.38 LegStruct.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/bom/LegDate.hpp>
00010 // AirTSP
00011 #include <airtsp/bom/LegStruct.hpp>
00012
00013 namespace AIRTSP {
00014
00015 // //////////////////////////////////////
00016 LegStruct::LegStruct ()
00017 : _boardingDateOffset (stdair::DEFAULT_DATE_OFFSET),
00018   _offDateOffset (stdair::DEFAULT_DATE_OFFSET) {
00019 }
00020
00021 // //////////////////////////////////////
00022 const std::string LegStruct::describe() const {
00023     std::ostringstream ostr;
00024     ostr << " " << _boardingPoint << " / "
00025           << boost::posix_time::to_simple_string(_boardingTime);
00026     if (_boardingDateOffset.days() != 0) {
00027         ostr << " [" << _boardingDateOffset.days() << "]";
00028     }
00029     ostr << " -- " << _offPoint << " / "
00030           << boost::posix_time::to_simple_string(_offTime);
00031     if (_offDateOffset.days() != 0) {
00032         ostr << " [" << _offDateOffset.days() << "]";
00033     }
00034     ostr << " --> "
00035           << boost::posix_time::to_simple_string(_elapsed)
00036           << std::endl;
00037     for (LegCabinStructList_T::const_iterator itCabin = _cabinList.begin();
00038          itCabin != _cabinList.end(); itCabin++) {
00039         const LegCabinStruct& lCabin = *itCabin;
00040         ostr << lCabin.describe();
00041     }
00042     ostr << std::endl;
00043     return ostr.str();
00044 }
00045
00046 // //////////////////////////////////////
00047 void LegStruct::fill (const stdair::Date_T& iRefDate,
00048                      stdair::LegDate& ioLegDate) const {
00049     // Set the Off Point
00050     ioLegDate.setOffPoint (_offPoint);
00051
00052     // Set the Boarding Date
00053     ioLegDate.setBoardingDate (iRefDate + _boardingDateOffset);
00054
00055     // Set the Boarding Time
00056     ioLegDate.setBoardingTime (_boardingTime);
00057
00058     // Set the Off Date
00059     ioLegDate.setOffDate (iRefDate + _offDateOffset);
00060
00061     // Set the Off Time
00062     ioLegDate.setOffTime (_offTime);
00063
00064     // Set the Elapsed Time
00065

```

```
00066     ioLegDate.setElapsedTime (_elapsed);
00067
00068     // Set the operating airline code
00069     ioLegDate.setOperatingAirlineCode (_airlineCode);
00070
00071     // Set the operating flight number
00072     ioLegDate.setOperatingFlightNumber (_flightNumber);
00073
00074 }
00075
00076 }
```



## 26.39 airtsp/bom/LegStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airtsp/bom/LegCabinStruct.hpp>
```

### Classes

- struct [AIRTSP::LegStruct](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

### Typedefs

- typedef std::vector< LegStruct > [AIRTSP::LegStructList\\_T](#)

## 26.40 LegStruct.hpp

```

00001 #ifndef __AIRTSP_BOM_LEGSTRUCT_HPP
00002 #define __AIRTSP_BOM_LEGSTRUCT_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/basic/StructAbstract.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/LegCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class LegDate;
00019 }
00020
00021 namespace AIRTSP {
00022
00023     struct LegStruct : public stdair::StructAbstract {
00024         // Attributes
00025         stdair::AirlineCode_T _airlineCode;
00026         stdair::FlightNumber_T _flightNumber;
00027         stdair::AirportCode_T _boardingPoint;
00028         stdair::DateOffset_T _boardingDateOffset;
00029         stdair::Duration_T _boardingTime;
00030         stdair::AirportCode_T _offPoint;
00031         stdair::DateOffset_T _offDateOffset;
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;
00037
00038         const std::string describe() const;
00039
00040         LegStruct();
00041     };
00042
00043     typedef std::vector<LegStruct> LegStructList_T;
00044 }
00045
00046 #endif // __AIRTSP_BOM_LEGSTRUCT_HPP

```

## 26.41 airtsp/bom/OnDPeriodStruct.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/bom/OnDPeriodStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.42 OnDPeriodStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <iostream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // AirTSP
00013 #include <airtsp/bom/OnDPeriodStruct.hpp>
00014
00015 namespace AIRTSP {
00016 // //////////////////////////////////////
00017 OnDPeriodStruct::OnDPeriodStruct ()
00018 : _datePeriod (stdair::BOOST_DEFAULT_DATE_PERIOD),
00019   _timeRangeStart (stdair::NULL_BOOST_TIME_DURATION),
00020   _timeRangeEnd (stdair::NULL_BOOST_TIME_DURATION),
00021   _nbOfAirlines (stdair::DEFAULT_NBFAIRLINES),
00022   _airlineCode (stdair::DEFAULT_NULL_AIRLINE_CODE),
00023   _classCode (stdair::DEFAULT_NULL_CLASS_CODE),
00024   _itSeconds (0) {
00025 }
00026
00027 // //////////////////////////////////////
00028 stdair::Date_T OnDPeriodStruct::getDate() const {
00029     return stdair::Date_T (_itYear, _itMonth, _itDay);
00030 }
00031
00032 // //////////////////////////////////////
00033 stdair::Duration_T OnDPeriodStruct::getTime() const {
00034     return boost::posix_time::hours (_itHours)
00035         + boost::posix_time::minutes (_itMinutes)
00036         + boost::posix_time::seconds (_itSeconds);
00037 }
00038
00039 // //////////////////////////////////////
00040 const std::string OnDPeriodStruct::describe() const {
00041     std::ostringstream ostr;
00042     ostr << _origin << "-" << _destination << ", "
00043         << _datePeriod << ", between "
00044         << boost::posix_time::to_simple_string(_timeRangeStart)
00045         << " to "
00046         << boost::posix_time::to_simple_string(_timeRangeEnd) << ", "
00047         << _classCode << ", "
00048         << _airlineCode << ", "
00049         << std::endl;
00050
00051     return ostr.str();
00052 }
00053
00054 // //////////////////////////////////////
00055 const std::string OnDPeriodStruct::describeTSKey() const {
00056     std::ostringstream ostr;
00057     ostr << _origin << "-" << _destination << ", "
00058         << _airlineCode << ", " << _classCode << std::endl;
00059
00060     return ostr.str();
00061 }
00062
00063 // //////////////////////////////////////
00064 const stdair::AirlineCode_T& OnDPeriodStruct::getFirstAirlineCode () const {
00065     assert (_airlineCodeList.size() > 0);

```

```
00066     stdair::AirlineCodeList_T::const_iterator itFirstAirlineCode =
00067         _airlineCodeList.begin();
00068     return *itFirstAirlineCode;
00069 }
00070
00071 }
```

## 26.43 airtsp/bom/OnDPeriodStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

### Classes

- struct [AIRTSP::OnDPeriodStruct](#)

### Namespaces

- namespace [AIRTSP](#)

## 26.44 OnDPeriodStruct.hpp

```

00001 #ifndef __AIRTSP_BOM_ONDPERIODSTRUCT_HPP
00002 #define __AIRTSP_BOM_ONDPERIODSTRUCT_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
00013 namespace AIRTSP {
00014
00016     struct OnDPeriodStruct : public stdair::StructAbstract {
00017     public:
00018         // ////////////////////////////////// Getters //////////////////////////////////
00020         const stdair::AirlineCode_T& getFirstAirlineCode () const;
00021
00023         stdair::Date_T getDate() const;
00024
00026         stdair::Duration_T getTime() const;
00027
00028         // ////////////////////////////////// Display Methods //////////////////////////////////
00030         const std::string describe() const;
00031
00034         const std::string describeTSKey() const;
00035
00036     public:
00038         OnDPeriodStruct ();
00039
00040     public:
00041         // Attributes
00042         stdair::AirportCode_T _origin;
00043         stdair::AirportCode_T _destination;
00044         stdair::DatePeriod_T _datePeriod;
00045         stdair::Duration_T _timeRangeStart;
00046         stdair::Duration_T _timeRangeEnd;
00047         stdair::NbOfAirlines_T _nbOfAirlines;
00048         stdair::AirlineCode_T _airlineCode;
00049         stdair::ClassCode_T _classCode;
00050         stdair::AirlineCodeList_T _airlineCodeList;
00051         stdair::ClassCodeList_T _classCodeList;
00052
00054         stdair::Date_T _dateRangeStart;
00055         stdair::Date_T _dateRangeEnd;
00056         unsigned int _itYear;
00057         unsigned int _itMonth;
00058         unsigned int _itDay;
00059
00061         long _itHours;
00062         long _itMinutes;
00063         long _itSeconds;
00064     };
00065 }
00066 #endif // __AIRTSP_BOM_ONDPERIODSTRUCT_HPP

```

## 26.45 airtsp/bom/OriginDestinationSet.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <airtsp/bom/OriginDestinationSet.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::OriginDestinationSet::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::OriginDestinationSet::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)



## 26.46 OriginDestinationSet.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/OriginDestinationSet.hpp>
00015
00016 namespace AIRTSP {
00017
00018     // //////////////////////////////////////
00019     OriginDestinationSet::OriginDestinationSet()
00020         : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
00021         assert (false);
00022     }
00023
00024     // //////////////////////////////////////
00025     OriginDestinationSet::OriginDestinationSet (const OriginDestinationSet&)
00026         : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
00027         assert (false);
00028     }
00029
00030     // //////////////////////////////////////
00031     OriginDestinationSet::OriginDestinationSet (const Key_T& iKey)
00032         : _key (iKey), _parent (NULL) {
00033     }
00034
00035     // //////////////////////////////////////
00036     OriginDestinationSet::~OriginDestinationSet () {
00037     }
00038
00039     // //////////////////////////////////////
00040     std::string OriginDestinationSet::toString() const {
00041         std::ostringstream oStr;
00042         oStr << _key.toString();
00043         return oStr.str();
00044     }
00045
00046     // //////////////////////////////////////
00047     void OriginDestinationSet::serialisationImplementationExport() const {
00048         std::ostringstream oStr;
00049         boost::archive::text_oarchive oa (oStr);
00050         oa << *this;
00051     }
00052
00053     // //////////////////////////////////////
00054     void OriginDestinationSet::serialisationImplementationImport() {
00055         std::istringstream iStr;
00056         boost::archive::text_iarchive ia (iStr);
00057         ia >> *this;
00058     }
00059
00060     // //////////////////////////////////////
00061     template<class Archive>
00062     void OriginDestinationSet::serialize (Archive& ioArchive,
00063         const unsigned int iFileVersion) {
00064         ioArchive & _key;
00065     }

```

```
00066
00067 // //////////////////////////////////////
00068 // Explicit template instantiation
00069 namespace ba = boost::archive;
00070 template
00071 void OriginDestinationSet::serialize<ba::text_oarchive> (ba::text_oarchive&,
00072                                                         unsigned int);
00073 template
00074 void OriginDestinationSet::serialize<ba::text_iarchive> (ba::text_iarchive&,
00075                                                         unsigned int);
00076 // //////////////////////////////////////
00077
00078 }
00079
```

## 26.47 airtsp/bom/OriginDestinationSet.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <airtsp/bom/OriginDestinationSetKey.hpp>
#include <airtsp/bom/OriginDestinationSetTypes.hpp>
```

### Classes

- class [AIRTSP::OriginDestinationSet](#)  
*Class representing a simple sub-network.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.48 OriginDestinationSet.hpp

```

00001 #ifndef __AIRTSP_BOM_ORIGINDESTINATIONSET_HPP
00002 #define __AIRTSP_BOM_ORIGINDESTINATIONSET_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 // AirTSP
00013 #include <airtsp/bom/OriginDestinationSetKey.hpp>
00014 #include <airtsp/bom/OriginDestinationSetTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     template <typename BOM> class FacBom;
00024     class FacBomManager;
00025 }
00026
00027 namespace AIRTSP {
00028
00029     class OriginDestinationSet : public stdair::BomAbstract {
00030     public:
00031         template <typename BOM> friend class stdair::FacBom;
00032         friend class stdair::FacBomManager;
00033         friend class boost::serialization::access;
00034
00035         // ////////////////////////////////// Type definitions //////////////////////////////////
00036         typedef OriginDestinationSetKey Key_T;
00037
00038     public:
00039         // ////////////////////////////////// Getters //////////////////////////////////
00040         const Key_T& getKey() const {
00041             return _key;
00042         }
00043
00044         const stdair::AirportCode_T& getDestination() const {
00045             return _key.getOffPoint();
00046         }
00047
00048         stdair::BomAbstract* const getParent() const {
00049             return _parent;
00050         }
00051
00052         const stdair::HolderMap_T& getHolderMap() const {
00053             return _holderMap;
00054         }
00055
00056     public:
00057         // ////////////////////////////////// Display support methods //////////////////////////////////
00058         void toStream (std::ostream& ioOut) const {
00059             ioOut << toString();
00060         }
00061
00062         void fromStream (std::istream& ioIn) {
00063
00064         }
00065     }
00066 }

```

```
00109
00113     std::string toString() const;
00114
00118     const std::string describeKey() const {
00119         return _key.toString();
00120     }
00121
00122
00123 public:
00124     // //////////// (Boost) Serialisation support methods ////////////
00128     template<class Archive>
00129     void serialize (Archive& ar, const unsigned int iFileVersion);
00130
00131 private:
00136     void serialisationImplementationExport() const;
00137     void serialisationImplementationImport();
00138
00139
00140 protected:
00141     // //////////// Constructors and destructors ////////////
00145     OriginDestinationSet (const Key_T&);
00146
00150     ~OriginDestinationSet ();
00151
00152 private:
00156     OriginDestinationSet ();
00157
00161     OriginDestinationSet (const OriginDestinationSet&);
00162
00163 protected:
00164     // //////////// Attributes ////////////
00168     Key_T _key;
00169
00173     stdair::BomAbstract* _parent;
00174
00178     stdair::HolderMap_T _holderMap;
00179 };
00180
00181 }
00182 #endif // __AIRTSP_BOM_ORIGINDESTINATIONSET_HPP
00183
```

## 26.49 airtsp/bom/OriginDestinationSetKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <airtsp/bom/OriginDestinationSetKey.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::OriginDestinationSetKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::OriginDestinationSetKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 26.50 OriginDestinationSetKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/OriginDestinationSetKey.hpp>
00015
00016 namespace AIRTSP {
00017
00018     // //////////////////////////////////////
00019     OriginDestinationSetKey::OriginDestinationSetKey()
00020         : _destination (stdair::DEFAULT_DESTINATION) {
00021         assert (false);
00022     }
00023
00024     // //////////////////////////////////////
00025     OriginDestinationSetKey::
00026     OriginDestinationSetKey (const stdair::AirportCode_T& iDestination)
00027         : _destination (iDestination) {
00028     }
00029
00030     // //////////////////////////////////////
00031     OriginDestinationSetKey::
00032     OriginDestinationSetKey (const OriginDestinationSetKey& iKey)
00033         : _destination (iKey._destination) {
00034     }
00035
00036     // //////////////////////////////////////
00037     OriginDestinationSetKey::~OriginDestinationSetKey() {
00038     }
00039
00040     // //////////////////////////////////////
00041     void OriginDestinationSetKey::toStream (std::ostream& ioOut) const {
00042         ioOut << "OriginDestinationSetKey: " << toString() << std::endl;
00043     }
00044
00045     // //////////////////////////////////////
00046     void OriginDestinationSetKey::fromStream (std::istream& ioIn) {
00047     }
00048
00049     // //////////////////////////////////////
00050     const std::string OriginDestinationSetKey::toString() const {
00051         std::ostringstream oStr;
00052         oStr << _destination;
00053         return oStr.str();
00054     }
00055
00056     // //////////////////////////////////////
00057     void OriginDestinationSetKey::serialisationImplementationExport() const {
00058         std::ostringstream oStr;
00059         boost::archive::text_oarchive oa (oStr);
00060         oa << *this;
00061     }
00062
00063     // //////////////////////////////////////
00064     void OriginDestinationSetKey::serialisationImplementationImport() {
00065         std::istringstream iStr;

```

```
00066     boost::archive::text_iarchive ia (iStr);
00067     ia >> *this;
00068 }
00069
00070 // //////////////////////////////////////
00071 template<class Archive>
00072 void OriginDestinationSetKey::serialize (Archive& ioArchive,
00073                                         const unsigned int iFileVersion) {
00074     ioArchive & _destination;
00075 }
00076
00077 // //////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template
00081 void OriginDestinationSetKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00082                                                             unsigned int);
00083 template
00084 void OriginDestinationSetKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00085                                                             unsigned int);
00086
00087 // //////////////////////////////////////
00088
00089
00090
00091
00092 }
```



## 26.51 airtsp/bom/OriginDestinationSetKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

### Classes

- struct [AIRTSP::OriginDestinationSetKey](#)  
*Structure representing the key of a sub-network.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [AIRTSP](#)

## 26.52 OriginDestinationSetKey.hpp

```

00001 #ifndef __AIRTSP_BOM_ORIGINDESTINATIONSETKEY_HPP
00002 #define __AIRTSP_BOM_ORIGINDESTINATIONSETKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace AIRTSP {
00022
00030     struct OriginDestinationSetKey : public stdair::KeyAbstract {
00031         friend class boost::serialization::access;
00032
00033         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00034     private:
00038         OriginDestinationSetKey();
00039
00040     public:
00044         OriginDestinationSetKey (const stdair::AirportCode_T& iDestination);
00045
00049         OriginDestinationSetKey (const OriginDestinationSetKey&);
00050
00054         ~OriginDestinationSetKey();
00055
00056     public:
00058         // ////////////////////////////////// Getters //////////////////////////////////
00062         const stdair::AirportCode_T& getOffPoint() const {
00063             return _destination;
00064         }
00065
00066     public:
00068         // ////////////////////////////////// Display support methods //////////////////////////////////
00074         void toStream (std::ostream& ioOut) const;
00075
00081         void fromStream (std::istream& ioIn);
00082
00092         const std::string toString() const;
00093
00094     public:
00096         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00100         template<class Archive>
00101         void serialize (Archive& ar, const unsigned int iFileVersion);
00102
00103     private:
00108         void serialisationImplementationExport() const;
00109         void serialisationImplementationImport();
00110
00111     private:
00112         // ////////////////////////////////// Attributes //////////////////////////////////
00113         stdair::AirportCode_T _destination;

```

```
00118     };  
00119  
00120 }  
00121 #endif // __AIRTSP_BOM_ORIGINDESTINATIONSETKEY_HPP
```

## 26.53 airtsp/bom/OriginDestinationSetTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef std::list< OriginDestinationSet \* > [AIRTSP::OriginDestinationSetList\\_T](#)
- typedef std::map< const stdair::MapKey\_T, OriginDestinationSet \* > [AIRTSP::OriginDestinationSetMap\\_T](#)

## 26.54 OriginDestinationSetTypes.hpp

```
00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __AIRTSP_BOM_ORIGINDESTINATIONSETTYPES_HPP
00003 #define __AIRTSP_BOM_ORIGINDESTINATIONSETTYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014
00015 namespace AIRTSP {
00016
00017     // Forward declarations.
00018     class OriginDestinationSet;
00019
00021     typedef std::list<OriginDestinationSet*> OriginDestinationSetList_T;
00022
00024     typedef std::map<const stdair::MapKey_T,
00025                    OriginDestinationSet*> OriginDestinationSetMap_T;
00026
00027 }
00028 #endif // __AIRTSP_BOM_ORIGINDESTINATIONSETTYPES_HPP
00029
```

## 26.55 airtsp/bom/ReachableUniverse.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <airtsp/bom/ReachableUniverse.hpp>
#include <airtsp/bom/SegmentPathPeriod.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::ReachableUniverse::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::ReachableUniverse::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 26.56 ReachableUniverse.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/ReachableUniverse.hpp>
00015 #include <airtsp/bom/SegmentPathPeriod.hpp>
00016
00017 namespace AIRTSP {
00018
00019 // //////////////////////////////////////
00020 ReachableUniverse::ReachableUniverse()
00021 : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
00022     assert (false);
00023 }
00024
00025 // //////////////////////////////////////
00026 ReachableUniverse::ReachableUniverse (const ReachableUniverse&)
00027 : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
00028     assert (false);
00029 }
00030
00031 // //////////////////////////////////////
00032 ReachableUniverse::ReachableUniverse (const Key_T& iKey)
00033 : _key (iKey), _parent (NULL) {
00034 }
00035
00036 // //////////////////////////////////////
00037 ReachableUniverse::~ReachableUniverse() {
00038 }
00039
00040 // //////////////////////////////////////
00041 std::string ReachableUniverse::toString() const {
00042     std::ostringstream oStr;
00043     oStr << _key.toString();
00044     return oStr.str();
00045 }
00046
00047 // //////////////////////////////////////
00048 void ReachableUniverse::serialisationImplementationExport() const {
00049     std::ostringstream oStr;
00050     boost::archive::text_oarchive oa (oStr);
00051     oa << *this;
00052 }
00053
00054 // //////////////////////////////////////
00055 void ReachableUniverse::serialisationImplementationImport() {
00056     std::istringstream iStr;
00057     boost::archive::text_iarchive ia (iStr);
00058     ia >> *this;
00059 }
00060
00061 // //////////////////////////////////////
00062 template<class Archive>
00063 void ReachableUniverse::serialize (Archive& ioArchive,
00064                                     const unsigned int iFileVersion) {
00065     ioArchive & _key;

```

```
00066     }
00067
00068     // //////////////////////////////////////
00069     // Explicit template instantiation
00070     namespace ba = boost::archive;
00071     template
00072     void ReachableUniverse::serialize<ba::text_oarchive> (ba::text_oarchive&,
00073                                                         unsigned int);
00074     template
00075     void ReachableUniverse::serialize<ba::text_iarchive> (ba::text_iarchive&,
00076                                                         unsigned int);
00077     // //////////////////////////////////////
00078
00079 }
00080
```



## 26.57 airtsp/bom/ReachableUniverse.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <airtsp/bom/ReachableUniverseKey.hpp>
#include <airtsp/bom/ReachableUniverseTypes.hpp>
#include <airtsp/bom/SegmentPathPeriodTypes.hpp>
```

### Classes

- class [AIRTSP::ReachableUniverse](#)  
*Class representing the root of the schedule-related BOM tree.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

**26.58 ReachableUniverse.hpp**

```

00001 #ifndef __AIRTSP_BOM_REACHABLEUNIVERSE_HPP
00002 #define __AIRTSP_BOM_REACHABLEUNIVERSE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 // AirTSP
00013 #include <airtsp/bom/ReachableUniverseKey.hpp>
00014 #include <airtsp/bom/ReachableUniverseTypes.hpp>
00015 #include <airtsp/bom/SegmentPathPeriodTypes.hpp>
00016
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00023 namespace stdair {
00024     template <typename BOM> class FacBom;
00025     class FacBomManager;
00026 }
00027
00028 namespace AIRTSP {
00029
00030     class ReachableUniverse : public stdair::BomAbstract {
00031     public:
00032         template <typename BOM> friend class stdair::FacBom;
00033         friend class stdair::FacBomManager;
00034         friend class SegmentPathGenerator;
00035         friend class boost::serialization::access;
00036
00037         // ////////////////////////////////// Type definitions //////////////////////////////////
00038         typedef ReachableUniverseKey Key_T;
00039
00040         // ////////////////////////////////// Getters //////////////////////////////////
00041         const Key_T& getKey() const {
00042             return _key;
00043         }
00044
00045         const stdair::AirportCode_T& getOrigin() const {
00046             return _key.getBoardingPoint();
00047         }
00048
00049         stdair::BomAbstract* getParent() const {
00050             return _parent;
00051         }
00052
00053         const stdair::HolderMap_T& getHolderMap() const {
00054             return _holderMap;
00055         }
00056
00057         const SegmentPathPeriodListList_T& getSegmentPathPeriodListList() const {
00058             return _segmentPathPeriodListList;
00059         }
00060
00061     public:
00062         // ////////////////////////////////// Display support methods //////////////////////////////////
00063         void toStream (std::ostream& ioOut) const {

```

```

00104         ioOut << toString();
00105     }
00106
00112     void fromStream (std::istream& ioIn) {
00113     }
00114
00118     std::string toString() const;
00119
00123     const std::string describeKey() const {
00124         return _key.toString();
00125     }
00126
00127
00128 public:
00129     // //////////// (Boost) Serialisation support methods ////////////
00133     template<class Archive>
00134     void serialize (Archive& ar, const unsigned int iFileVersion);
00135
00136 private:
00141     void serialisationImplementationExport() const;
00142     void serialisationImplementationImport();
00143
00144
00145 protected:
00146     // //////////// Constructors and destructors ////////////
00150     ReachableUniverse (const Key_T&);
00151
00155     ~ReachableUniverse();
00156
00157 private:
00161     ReachableUniverse();
00162
00166     ReachableUniverse (const ReachableUniverse&);
00167
00168
00169 protected:
00170     // //////////// Attributes ////////////
00174     Key_T _key;
00175
00179     stdair::BomAbstract* _parent;
00180
00184     stdair::HolderMap_T _holderMap;
00185
00191     SegmentPathPeriodListList_T _segmentPathPeriodListList;
00192 };
00193
00194 }
00195 #endif // __AIRTSP_BOM_REACHABLEUNIVERSE_HPP
00196

```

## 26.59 airtsp/bom/ReachableUniverseKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <airtsp/bom/ReachableUniverseKey.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::ReachableUniverseKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::ReachableUniverseKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 26.60 ReachableUniverseKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/ReachableUniverseKey.hpp>
00015
00016 namespace AIRTSP {
00017
00018     // //////////////////////////////////////
00019     ReachableUniverseKey::ReachableUniverseKey()
00020         : _origin (stdair::DEFAULT_ORIGIN) {
00021         assert (false);
00022     }
00023
00024     // //////////////////////////////////////
00025     ReachableUniverseKey::
00026     ReachableUniverseKey (const ReachableUniverseKey& iKey)
00027         : _origin (iKey._origin) {
00028     }
00029
00030     // //////////////////////////////////////
00031     ReachableUniverseKey::
00032     ReachableUniverseKey (const stdair::AirportCode_T& iAirportCode)
00033         : _origin (iAirportCode) {
00034     }
00035
00036     // //////////////////////////////////////
00037     ReachableUniverseKey::~ReachableUniverseKey() {
00038     }
00039
00040     // //////////////////////////////////////
00041     void ReachableUniverseKey::toStream (std::ostream& ioOut) const {
00042         ioOut << "ReachableUniverseKey: " << toString() << std::endl;
00043     }
00044
00045     // //////////////////////////////////////
00046     void ReachableUniverseKey::fromStream (std::istream& ioIn) {
00047     }
00048
00049     // //////////////////////////////////////
00050     const std::string ReachableUniverseKey::toString() const {
00051         std::ostringstream oStr;
00052         oStr << _origin;
00053         return oStr.str();
00054     }
00055
00056     // //////////////////////////////////////
00057     void ReachableUniverseKey::serialisationImplementationExport() const {
00058         std::ostringstream oStr;
00059         boost::archive::text_oarchive oa (oStr);
00060         oa << *this;
00061     }
00062
00063     // //////////////////////////////////////
00064     void ReachableUniverseKey::serialisationImplementationImport() {
00065         std::istringstream iStr;

```

```
00066     boost::archive::text_iarchive ia (iStr);
00067     ia >> *this;
00068 }
00069
00070 // //////////////////////////////////////
00071 template<class Archive>
00072 void ReachableUniverseKey::serialize (Archive& ioArchive,
00073                                       const unsigned int iFileVersion) {
00074     ioArchive & _origin;
00075 }
00076
00077 // //////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template
00081 void ReachableUniverseKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00082                                                         unsigned int);
00083 template
00084 void ReachableUniverseKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00085                                                         unsigned int);
00086 // //////////////////////////////////////
00087
00088 }
```

## 26.61 airtsp/bom/ReachableUniverseKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

### Classes

- struct [AIRTSP::ReachableUniverseKey](#)  
*Structure representing the key of the schedule-related BOM tree root.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [AIRTSP](#)

## 26.62 ReachableUniverseKey.hpp

```

00001 #ifndef __AIRTSP_BOM_REACHABLEUNIVERSEKEY_HPP
00002 #define __AIRTSP_BOM_REACHABLEUNIVERSEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace AIRTSP {
00022
00033     struct ReachableUniverseKey : public stdair::KeyAbstract {
00034         friend class boost::serialization::access;
00035
00036         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00037     private:
00041         ReachableUniverseKey();
00042
00043     public:
00047         ReachableUniverseKey (const stdair::AirportCode_T& iOrigin);
00048
00052         ReachableUniverseKey (const ReachableUniverseKey&);
00053
00057         ~ReachableUniverseKey();
00058
00059     public:
00061         // ////////////////////////////////// Getters //////////////////////////////////
00066         const stdair::AirportCode_T& getBoardingPoint() const {
00067             return _origin;
00068         }
00069
00070     public:
00072         // ////////////////////////////////// Display support methods //////////////////////////////////
00078         void toStream (std::ostream& ioOut) const;
00079
00085         void fromStream (std::istream& ioIn);
00086
00096         const std::string toString() const;
00097
00098     public:
00100         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00104         template<class Archive>
00105         void serialize (Archive& ar, const unsigned int iFileVersion);
00106
00107     private:
00112         void serialisationImplementationExport() const;
00113         void serialisationImplementationImport();
00114
00115     private:
00117         // ////////////////////////////////// Attributes //////////////////////////////////
00122         stdair::AirportCode_T _origin;

```



```
00123     };  
00124  
00125 }  
00126  
00127 #endif // __AIRTSP_BOM_REACHABLEUNIVERSEKEY_HPP
```

## 26.63 airtsp/bom/ReachableUniverseTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef std::list< ReachableUniverse \* > [AIRTSP::ReachableUniverseList\\_T](#)
- typedef std::map< const stdair::MapKey\_T, ReachableUniverse \* >  
[AIRTSP::ReachableUniverseMap\\_T](#)

## 26.64 ReachableUniverseTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __AIRTSP_BOM_REACHABLEUNIVERSETYPES_HPP
00003 #define __AIRTSP_BOM_REACHABLEUNIVERSETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014
00015 namespace AIRTSP {
00016
00017     // Forward declarations.
00018     class ReachableUniverse;
00019
00021     typedef std::list<ReachableUniverse*> ReachableUniverseList_T;
00022
00024     typedef std::map<const stdair::MapKey_T,
00025                    ReachableUniverse*> ReachableUniverseMap_T;
00026
00027 }
00028 #endif // __AIRTSP_BOM_REACHABLEUNIVERSETYPES_HPP
00029
```

## 26.65 airtsp/bom/SegmentCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentCabin.hpp>
#include <airtsp/bom/SegmentCabinStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.66 SegmentCabinStruct.cpp**

```
00001 // ////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/SegmentCabin.hpp>
00009 // AirTSP
00010 #include <airtsp/bom/SegmentCabinStruct.hpp>
00011
00012 namespace AIRTSP {
00013
00014 // ////////////////////////////////////////
00015 const std::string SegmentCabinStruct::describe() const {
00016     std::ostringstream ostr;
00017     ostr << "          " << _cabinCode << " " << _classes;
00018     return ostr.str();
00019 }
00020
00021 // ////////////////////////////////////////
00022 void SegmentCabinStruct::fill (stdair::SegmentCabin& ioSegmentCabin) const {
00023 }
00024
00025 }
```

## 26.67 airtsp/bom/SegmentCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airtsp/bom/FareFamilyStruct.hpp>
```

### Classes

- struct [AIRTSP::SegmentCabinStruct](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

### Typedefs

- typedef std::vector< SegmentCabinStruct > [AIRTSP::SegmentCabinStructList\\_T](#)

**26.68 SegmentCabinStruct.hpp**

```

00001 #ifndef __AIRTSP_BOM_SEGMENTCABINSTRUCT_HPP
00002 #define __AIRTSP_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 // AirTSP
00014 #include <airtsp/bom/FareFamilyStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentCabin;
00019 }
00020
00021 namespace AIRTSP {
00022
00024     struct SegmentCabinStruct : public stdair::StructAbstract {
00025         // Attributes
00026         stdair::CabinCode_T _cabinCode;
00027         stdair::ClassList_String_T _classes;
00028         stdair::FamilyCode_T _itFamilyCode;
00029         stdair::CurveKey_T _itFRAT5CurveKey;
00030         stdair::CurveKey_T _itFFDisutilityCurveKey;
00031         FareFamilyStructList_T _fareFamilies;
00032
00033         void fill (stdair::SegmentCabin&) const;
00034
00035         const std::string describe() const;
00036
00037     };
00038
00039     typedef std::vector<SegmentCabinStruct> SegmentCabinStructList_T;
00040
00041 }
00042 #endif // __AIRTSP_BOM_SEGMENTCABINSTRUCT_HPP

```

## 26.69 airtsp/bom/SegmentPathPeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightPeriod.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/BomManager.hpp>
#include <airtsp/bom/SegmentPathPeriod.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::SegmentPathPeriod::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::SegmentPathPeriod::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)



## 26.70 SegmentPathPeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasConst_Period_BOM.hpp>
00015 #include <stdair/basic/BasConst_TravelSolution.hpp>
00016 #include <stdair/bom/Inventory.hpp>
00017 #include <stdair/bom/FlightPeriod.hpp>
00018 #include <stdair/bom/SegmentPeriod.hpp>
00019 #include <stdair/bom/BomManager.hpp>
00020 // AirTSP
00021 #include <airtsp/bom/SegmentPathPeriod.hpp>
00022
00023 namespace AIRTSP {
00024
00025 // //////////////////////////////////////
00026 SegmentPathPeriod::SegmentPathPeriod()
00027 : _key (stdair::PeriodStruct (stdair::BOOST_DEFAULT_DATE_PERIOD,
00028                               stdair::DEFAULT_DOW_STRING),
00029         stdair::NULL_BOOST_TIME_DURATION, stdair::NULL_BOOST_TIME_DURATION,
00030         DateOffsetList_T(),
00031         stdair::DEFAULT_NBOFAIRLINES),
00032   _parent (NULL) {
00033     assert (false);
00034 }
00035
00036 // //////////////////////////////////////
00037 SegmentPathPeriod::SegmentPathPeriod (const SegmentPathPeriod& iSPP)
00038 : _key (iSPP._key), _parent (NULL) {
00039     assert (false);
00040 }
00041
00042 // //////////////////////////////////////
00043 SegmentPathPeriod::SegmentPathPeriod (const Key_T& iKey)
00044 : _key (iKey), _parent (NULL) {
00045 }
00046
00047 // //////////////////////////////////////
00048 SegmentPathPeriod::~SegmentPathPeriod() {
00049 }
00050
00051 // //////////////////////////////////////
00052 std::string SegmentPathPeriod::toString() const {
00053     std::ostringstream oStr;
00054     oStr << _key.toString();
00055     return oStr.str();
00056 }
00057
00058 // //////////////////////////////////////
00059 void SegmentPathPeriod::serialisationImplementationExport() const {
00060     std::ostringstream oStr;
00061     boost::archive::text_oarchive oa (oStr);
00062     oa << *this;
00063 }
00064
00065 // //////////////////////////////////////

```

```

00066 void SegmentPathPeriod::serialisationImplementationImport() {
00067     std::istringstream iStr;
00068     boost::archive::text_iarchive ia (iStr);
00069     ia >> *this;
00070 }
00071
00072 // //////////////////////////////////////
00073 template<class Archive>
00074 void SegmentPathPeriod::serialize (Archive& ioArchive,
00075                                     const unsigned int iFileVersion) {
00076     ioArchive & _key;
00077 }
00078
00079 // //////////////////////////////////////
00080 // Explicit template instantiation
00081 namespace ba = boost::archive;
00082 template
00083 void SegmentPathPeriod::serialize<ba::text_oarchive> (ba::text_oarchive&,
00084                                                         unsigned int);
00085 template
00086 void SegmentPathPeriod::serialize<ba::text_iarchive> (ba::text_iarchive&,
00087                                                         unsigned int);
00088 // //////////////////////////////////////
00089
00090 // //////////////////////////////////////
00091 stdair::SegmentPeriod* SegmentPathPeriod::getLastSegmentPeriod () const {
00092     // Retrieve the last segment of the list
00093     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00094         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00095     stdair::SegmentPeriodList_T::const_reverse_iterator itLastSegment =
00096         lSegmentPeriodList.rbegin();
00097
00098     if (itLastSegment == lSegmentPeriodList.rend()) {
00099         return NULL;
00100     }
00101
00102     stdair::SegmentPeriod* oSegment_ptr = *itLastSegment;
00103     assert (oSegment_ptr != NULL);
00104
00105     return oSegment_ptr;
00106 }
00107
00108 // //////////////////////////////////////
00109 stdair::SegmentPeriod* SegmentPathPeriod::getFirstSegmentPeriod () const {
00110     // Retrieve the first segment of the list
00111     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00112         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00113     stdair::SegmentPeriodList_T::const_iterator itFirstSegment =
00114         lSegmentPeriodList.begin();
00115
00116     if (itFirstSegment == lSegmentPeriodList.end()) {
00117         return NULL;
00118     }
00119
00120     stdair::SegmentPeriod* oSegment_ptr = *itFirstSegment;
00121     assert (oSegment_ptr != NULL);
00122
00123     return oSegment_ptr;
00124 }
00125
00126 // //////////////////////////////////////
00127 const stdair::AirportCode_T& SegmentPathPeriod::getDestination () const {
00128     const stdair::SegmentPeriod* lLastSegment_ptr = getLastSegmentPeriod();
00129     assert (lLastSegment_ptr != NULL);
00130     return lLastSegment_ptr->getOffPoint();
00131 }
00132

```

```

00133 // //////////////////////////////////////
00134 bool SegmentPathPeriod::
00135 isAirlineFlown (const stdair::AirlineCode_T& iAirlineCode) const {
00136     bool oAirlineFlown = false;
00137
00138     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00139         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00140     for (stdair::SegmentPeriodList_T::const_iterator itSegmentPeriod =
00141         lSegmentPeriodList.begin();
00142         itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
00143         const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
00144         assert (lSegmentPeriod_ptr != NULL);
00145
00146         const stdair::FlightPeriod& lFlightPeriod =
00147             stdair::BomManager::getParent<stdair::FlightPeriod>(*lSegmentPeriod_ptr);
00148
00149         const stdair::Inventory& lInventory =
00150             stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00151         const stdair::AirlineCode_T& lSegmentAirlineCode =
00152             lInventory.getAirlineCode ();
00153         if (lSegmentAirlineCode == iAirlineCode) {
00154             oAirlineFlown = true;
00155             break;
00156         }
00157     }
00158     return oAirlineFlown;
00159 }
00160
00161 // //////////////////////////////////////
00162 SegmentPathPeriodKey SegmentPathPeriod::
00163 connectWithAnotherSegment(const SegmentPathPeriod& iSingleSegmentPath) const {
00164     SegmentPathPeriodKey oSegmentPathPeriodKey;
00165
00166     // Retrieve the (only) segment period of the single segment path.
00167     const stdair::SegmentPeriod* lNextSegmentPeriod_ptr =
00168         iSingleSegmentPath.getFirstSegmentPeriod();
00169     assert (lNextSegmentPeriod_ptr != NULL);
00170
00171     // Retrieve the last segment period of the current segment path and check
00172     // if the combination of the last segment and the next segment that we
00173     // want to add to the current segment path will create a new segment
00174     // (i.e., the two segment period belongs to the same flight number).
00175     const stdair::SegmentPeriod* lLastSegmentPeriod_ptr = getLastSegmentPeriod ()
;
00176     assert (lLastSegmentPeriod_ptr != NULL);
00177     const stdair::FlightPeriod& lLastFlightPeriod = stdair::BomManager::
00178         getParent<stdair::FlightPeriod> (*lLastSegmentPeriod_ptr);
00179     const stdair::Inventory& lLastInventory =
00180         stdair::BomManager::getParent<stdair::Inventory> (lLastFlightPeriod);
00181
00182     const stdair::FlightPeriod& lNextFlightPeriod = stdair::BomManager::
00183         getParent<stdair::FlightPeriod> (*lNextSegmentPeriod_ptr);
00184     const stdair::Inventory& lNextInventory =
00185         stdair::BomManager::getParent<stdair::Inventory> (lNextFlightPeriod);
00186
00187     if (lLastFlightPeriod.getFlightNumber() == lNextFlightPeriod.getFlightNumber()
00188         && lLastInventory.getAirlineCode() == lNextInventory.getAirlineCode()) {
00189         return oSegmentPathPeriodKey;
00190     }
00191
00192     // Check if the new segment period will create a circle.
00193     const stdair::AirportCode_T& lDestination =
00194         lNextSegmentPeriod_ptr->getOffPoint();
00195     if (checkCircle (lDestination) == true) {
00196         return oSegmentPathPeriodKey;
00197     }

```

```

00198
00199 // Check if a passenger can connect from the last segment of the
00200 // current segment path to the first segment of the to-be-added
00201 // segment path. If yes, build a new departure period for the new
00202 // segment path.
00203 DateOffsetList_T lBoardingDateOffsetList =
00204     getBoardingDateOffsetList();
00205 const stdair::PeriodStruct& lCurrentDeparturePeriod = getDeparturePeriod();
00206 const stdair::PeriodStruct& lNextDeparturePeriod =
00207     iSingleSegmentPath.getDeparturePeriod();
00208 const stdair::Duration_T& lLastOffTime =
00209     lLastSegmentPeriod_ptr->getOffTime();
00210 const stdair::Duration_T& lNextBoardingTime =
00211     lNextSegmentPeriod_ptr->getBoardingTime();
00212 // If the next boarding time is later than the last off time, check if
00213 // the passengers will have enough time for the transfer. If the next
00214 // boarding time is earlier than the last off time, check if the passengers
00215 // can connect to a flight in the next day.
00216 if (lNextBoardingTime >= lLastOffTime) {
00217     const stdair::Duration_T lStopTime = lNextBoardingTime - lLastOffTime;
00218     if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
00219         return oSegmentPathPeriodKey;
00220     } else {
00221         // Calculate the date offset of the next segment compare to
00222         // the first one. In this case, this value is equal to the offset
00223         // of the off date of the last segment compare to the boarding date
00224         // of the first segment.
00225         const stdair::DateOffset_T& lLastBoardingDateOffset =
00226             lBoardingDateOffsetList.at (getNbOfSegments() - 1);
00227         const stdair::DateOffset_T lNextBoardingDateOffset =
00228             lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
00229             - lLastSegmentPeriod_ptr->getBoardingDateOffset();
00230         const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
00231             stdair::DateOffset_T (0) - lNextBoardingDateOffset;
00232
00233         // Compute the adjusted departure period of the next segment by
00234         // subtracting the origin one with the boarding date offset.
00235         const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
00236             lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);
00237
00238         // Build the intersection of the two periods.
00239         const stdair::PeriodStruct lNewDeparturePeriod =
00240             lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
00241         stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
00242             lNextSegmentPeriod_ptr->getElapsedTime();
00243         lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
00244         oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
00245         oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
00246     }
00247 } else {
00248     const stdair::Duration_T lStopTime =
00249         lNextBoardingTime - lLastOffTime + stdair::Duration_T (24, 0, 0);
00250     if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
00251         return oSegmentPathPeriodKey;
00252     } else {
00253         // Calculate the date offset of the next segment compare to
00254         // the first one.
00255         const stdair::DateOffset_T& lLastBoardingDateOffset =
00256             lBoardingDateOffsetList.at (getNbOfSegments() - 1);
00257         const stdair::DateOffset_T lNextBoardingDateOffset =
00258             lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
00259             - lLastSegmentPeriod_ptr->getBoardingDateOffset() +
00260             stdair::DateOffset_T (1);
00261         const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
00262             stdair::DateOffset_T (0) - lNextBoardingDateOffset;
00263
00264         // Compute the adjusted departure period of the next segment by

```

```

00265         // subtracting the origin one with the boarding date offset.
00266         const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
00267             lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);
00268
00269         // Build the intersection of the two periods.
00270         const stdair::PeriodStruct lNewDeparturePeriod =
00271             lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
00272         stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
00273             lNextSegmentPeriod_ptr->getElapsedTime();
00274         lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
00275         oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
00276         oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
00277     }
00278 }
00279
00280 const stdair::Duration_T& lBoardingTime = getBoardingTime();
00281 oSegmentPathPeriodKey.setBoardingTime (lBoardingTime);
00282 oSegmentPathPeriodKey.setBoardingDateOffsetList (lBoardingDateOffsetList);
00283
00284 return oSegmentPathPeriodKey;
00285 }
00286
00287 // //////////////////////////////////////
00288 bool SegmentPathPeriod::
00289 checkCircle (const stdair::AirlineCode_T& iDestination) const {
00290     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00291         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00292     for (stdair::SegmentPeriodList_T::const_iterator itSegment =
00293         lSegmentPeriodList.begin();
00294         itSegment != lSegmentPeriodList.end(); ++itSegment) {
00295         const stdair::SegmentPeriod* lCurrentSegment_ptr = *itSegment;
00296         assert (lCurrentSegment_ptr != NULL);
00297         const stdair::AirlineCode_T& lCurrentBoardingPoint =
00298             lCurrentSegment_ptr->getBoardingPoint();
00299         if (lCurrentBoardingPoint == iDestination) {
00300             return true;
00301         }
00302     }
00303     return false;
00304 }
00305
00306 // //////////////////////////////////////
00307 bool SegmentPathPeriod::
00308 isDepartureDateValid (const stdair::Date_T& iDepartureDate) const {
00309     const stdair::PeriodStruct& lPeriod = getDeparturePeriod ();
00310
00311     // Check if the departure date is within the date range.
00312     const stdair::DatePeriod_T& lDeparturePeriod = lPeriod.getDateRange ();
00313     if (lDeparturePeriod.contains (iDepartureDate) == false) {
00314         return false;
00315     }
00316
00317     // Check if the departure date is valid within the DOW.
00318     // 0 = Sunday, 1 = Monday, etc.
00319     const short lDay = iDepartureDate.day_of_week ();
00320     const stdair::DoWStruct& lDoW = lPeriod.getDoW ();
00321     if (lDoW.getStandardDayOfWeek (lDay) == false) {
00322         return false;
00323     }
00324
00325     return true;
00326 }
00327
00328 }

```

## 26.71 airtsp/bom/SegmentPathPeriod.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <airtsp/bom/SegmentPathPeriodKey.hpp>
#include <airtsp/bom/SegmentPathPeriodTypes.hpp>
```

### Classes

- class [AIRTSP::SegmentPathPeriod](#)  
*Class representing a segment/path.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.72 SegmentPathPeriod.hpp

```

00001 #ifndef __AIRTSP_BOM_SEGMENTPATHPERIOD_HPP
00002 #define __AIRTSP_BOM_SEGMENTPATHPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 // AirTSP
00013 #include <airtsp/bom/SegmentPathPeriodKey.hpp>
00014 #include <airtsp/bom/SegmentPathPeriodTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     template <typename BOM> class FacBom;
00024     class FacBomManager;
00025     class SegmentPeriod;
00026 }
00027
00028 namespace AIRTSP {
00029
00030     class SegmentPathPeriod : public stdair::BomAbstract {
00031     public:
00032         template <typename BOM> friend class stdair::FacBom;
00033         friend class stdair::FacBomManager;
00034         friend class boost::serialization::access;
00035
00036         // ////////////////////////////////// Type definitions //////////////////////////////////
00037         typedef SegmentPathPeriodKey Key_T;
00038
00039     public:
00040         // ////////////////////////////////// Getters //////////////////////////////////
00041         const Key_T& getKey() const {
00042             return _key;
00043         }
00044
00045         stdair::BomAbstract* const getParent() const {
00046             return _parent;
00047         }
00048
00049         const stdair::PeriodStruct& getDeparturePeriod() const {
00050             return _key.getPeriod();
00051         }
00052
00053         const DateOffsetList_T& getBoardingDateOffsetList () const {
00054             return _key.getBoardingDateOffsetList();
00055         }
00056
00057         const stdair::NbOfSegments_T getNbOfSegments() const {
00058             return _key.getNbOfSegments();
00059         }
00060
00061         const stdair::NbOfAirlines_T& getNbOfAirlines() const {
00062             return _key.getNbOfAirlines();
00063         }
00064     }
00065 }

```

```

00092     const stdair::Duration_T& getElapsedTime() const {
00093         return _key.getElapsedTime();
00094     }
00095
00096     const stdair::Duration_T& getBoardingTime() const {
00097         return _key.getBoardingTime();
00098     }
00099
00100     const stdair::HolderMap_T& getHolderMap() const {
00101         return _holderMap;
00102     }
00103
00104     stdair::SegmentPeriod* getLastSegmentPeriod() const;
00105
00106     stdair::SegmentPeriod* getFirstSegmentPeriod() const;
00107
00108     const stdair::AirportCode_T& getDestination() const;
00109
00110 public:
00111     // //////////// Business methods ////////////
00112     Key_T connectWithAnotherSegment (const SegmentPathPeriod&) const;
00113
00114     bool checkCircle (const stdair::AirportCode_T&) const;
00115
00116     bool isAirlineFlown (const stdair::AirlineCode_T&) const;
00117
00118     bool isDepartureDateValid (const stdair::Date_T&) const;
00119
00120 public:
00121     // //////////// Display support methods ////////////
00122     void toStream (std::ostream& ioOut) const {
00123         ioOut << toString();
00124     }
00125
00126     void fromStream (std::istream& ioIn) {
00127     }
00128
00129     std::string toString() const;
00130
00131     const std::string describeKey() const {
00132         return _key.toString();
00133     }
00134
00135 public:
00136     // //////////// (Boost) Serialisation support methods ////////////
00137     template<class Archive>
00138     void serialize (Archive& ar, const unsigned int iFileVersion);
00139
00140 private:
00141     void serialisationImplementationExport() const;
00142     void serialisationImplementationImport();
00143
00144 protected:
00145     // //////////// Constructors and destructors ////////////
00146     SegmentPathPeriod (const Key_T&);
00147
00148     ~SegmentPathPeriod();
00149
00150 private:
00151     SegmentPathPeriod();
00152
00153     SegmentPathPeriod (const SegmentPathPeriod&);
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241

```



```
00242     protected:
00243         // ////////// Attributes //////////
00249         Key_T _key;
00250
00254         stdair::BomAbstract* _parent;
00255
00262         stdair::HolderMap_T _holderMap;
00263     };
00264
00265 }
00266 #endif // __AIRTSP_BOM_SEGMENTPATHPERIOD_HPP
00267
```

## 26.73 airtsp/bom/SegmentPathPeriodKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <airtsp/bom/SegmentPathPeriodKey.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Functions

- template void [AIRTSP::SegmentPathPeriodKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [AIRTSP::SegmentPathPeriodKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 26.74 SegmentPathPeriodKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasConst_Period_BOM.hpp>
00015 #include <stdair/basic/BasConst_TravelSolution.hpp>
00016 // AirTSP
00017 #include <airtsp/bom/SegmentPathPeriodKey.hpp>
00018
00019 namespace AIRTSP {
00020
00021 // //////////////////////////////////////
00022 SegmentPathPeriodKey::SegmentPathPeriodKey()
00023 : _period (stdair::BOOST_DEFAULT_DATE_PERIOD, stdair::DEFAULT_DOW_STRING),
00024   _boardingTime (stdair::NULL_BOOST_TIME_DURATION),
00025   _elapsed (stdair::NULL_BOOST_TIME_DURATION),
00026   _nbOfAirlines (stdair::DEFAULT_NBFAIRLINES) {
00027 }
00028
00029 // //////////////////////////////////////
00030 SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey& iSPPK)
00031 : _period (iSPPK._period),
00032   _boardingTime (iSPPK._boardingTime),
00033   _elapsed (iSPPK._elapsed),
00034   _boardingDateOffsetList (iSPPK._boardingDateOffsetList),
00035   _nbOfAirlines (iSPPK._nbOfAirlines) {
00036 }
00037
00038 // //////////////////////////////////////
00039 SegmentPathPeriodKey::
00040 SegmentPathPeriodKey (const stdair::PeriodStruct& iPeriod,
00041                      const stdair::Duration_T& iBoardingTime,
00042                      const stdair::Duration_T& iElapsedTime,
00043                      const DateOffsetList_T& iBoardingDateOffsetList,
00044                      const stdair::NbOfAirlines_T& iNbOfAirlines)
00045 : _period (iPeriod),
00046   _boardingTime (iBoardingTime),
00047   _elapsed (iElapsedTime),
00048   _boardingDateOffsetList (iBoardingDateOffsetList),
00049   _nbOfAirlines (iNbOfAirlines) {
00050 }
00051
00052 // //////////////////////////////////////
00053 SegmentPathPeriodKey::~SegmentPathPeriodKey() {
00054 }
00055
00056 // //////////////////////////////////////
00057 void SegmentPathPeriodKey::toStream (std::ostream& ioOut) const {
00058   ioOut << "SegmentPathPeriodKey: " << toString() << std::endl;
00059 }
00060
00061 // //////////////////////////////////////
00062 void SegmentPathPeriodKey::fromStream (std::istream& ioIn) {
00063 }
00064
00065 // //////////////////////////////////////

```

```

00066     const std::string SegmentPathPeriodKey::toString() const {
00067         std::ostringstream ostr;
00068         ostr << _period.describeShort() << ", "
00069             << _boardingTime << ", " << _elapsed << ", ";
00070
00071         for (DateOffsetList_T::const_iterator itOffset =
00072             _boardingDateOffsetList.begin();
00073             itOffset != _boardingDateOffsetList.end(); ++itOffset) {
00074             const stdair::DateOffset_T& lDateOffset = *itOffset;
00075             ostr << lDateOffset.days() << ", ";
00076         }
00077
00078         ostr << _nbOfAirlines ;
00079         return ostr.str();
00080     }
00081
00082     ///////////////////////////////////////////////////////////////////
00083     void SegmentPathPeriodKey::serialisationImplementationExport() const {
00084         std::ostringstream ostr;
00085         boost::archive::text_oarchive oa (ostr);
00086         oa << *this;
00087     }
00088
00089     ///////////////////////////////////////////////////////////////////
00090     void SegmentPathPeriodKey::serialisationImplementationImport() {
00091         std::istringstream istr;
00092         boost::archive::text_iarchive ia (istr);
00093         ia >> *this;
00094     }
00095
00096     ///////////////////////////////////////////////////////////////////
00097     template<class Archive>
00098     void SegmentPathPeriodKey::serialize (Archive& ioArchive,
00099                                         const unsigned int iFileVersion) {
00100         //ioArchive & _period & _boardingTime & _elapsed & _nbOfAirlines;
00101         std::string lBTStr = boost::posix_time::to_simple_string (_boardingTime);
00102         std::string lElapsedStr = boost::posix_time::to_simple_string (_elapsed);
00103         ioArchive & lBTStr & lElapsedStr & _nbOfAirlines;
00104     }
00105
00106     ///////////////////////////////////////////////////////////////////
00107     // Explicit template instantiation
00108     namespace ba = boost::archive;
00109     template
00110     void SegmentPathPeriodKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00111                                                             unsigned int);
00112     template
00113     void SegmentPathPeriodKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00114                                                             unsigned int);
00115     ///////////////////////////////////////////////////////////////////
00116
00117 }

```

## 26.75 airtsp/bom/SegmentPathPeriodKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>
#include <airtsp/bom/SegmentPathPeriodTypes.hpp>
```

### Classes

- struct [AIRTSP::SegmentPathPeriodKey](#)  
*Structure representing the key of a segment/path.*

### Namespaces

- namespace [boost](#)  
*Forward declarations.*
- namespace [boost::serialization](#)
- namespace [AIRTSP](#)

**26.76 SegmentPathPeriodKey.hpp**

```

00001 #ifndef __AIRTSP_BOM_SEGMENTPATHPERIODKEY_HPP
00002 #define __AIRTSP_BOM_SEGMENTPATHPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/bom/KeyAbstract.hpp>
00014 #include <stdair/bom/PeriodStruct.hpp>
00015 // AirTSP
00016 #include <airtsp/bom/SegmentPathPeriodTypes.hpp>
00017
00019 namespace boost {
00020     namespace serialization {
00021         class access;
00022     }
00023 }
00024
00025 namespace AIRTSP {
00026
00033     struct SegmentPathPeriodKey : public stdair::KeyAbstract {
00034         friend class boost::serialization::access;
00035
00036         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00037     public:
00041         SegmentPathPeriodKey (const stdair::PeriodStruct&,
00042                               const stdair::Duration_T& iBoardingTime,
00043                               const stdair::Duration_T& iElapsed,
00044                               const DateOffsetList_T&,
00045                               const stdair::NbOfAirlines_T&);
00046
00050         SegmentPathPeriodKey();
00051
00055         SegmentPathPeriodKey (const SegmentPathPeriodKey&);
00056
00060         ~SegmentPathPeriodKey();
00061
00062     public:
00064         // ////////////////////////////////// Getters //////////////////////////////////
00068         const stdair::PeriodStruct& getPeriod() const {
00069             return _period;
00070         }
00071
00075         const DateOffsetList_T& getBoardingDateOffsetList() const {
00076             return _boardingDateOffsetList;
00077         }
00078
00082         const stdair::NbOfSegments_T getNbOfSegments() const {
00083             return _boardingDateOffsetList.size();
00084         }
00085
00089         const stdair::NbOfAirlines_T& getNbOfAirlines() const {
00090             return _nbOfAirlines;
00091         }
00092
00096         const stdair::Duration_T& getElapsedTime() const {
00097             return _elapsed;
00098         }
00099

```

```
00103     const stdair::Duration_T& getBoardingTime() const {
00104         return _boardingTime;
00105     }
00106
00107
00108 public:
00109     // //////////// Setters ////////////
00111     void setPeriod (const stdair::PeriodStruct& iPeriod) {
00112         _period = iPeriod;
00113     }
00114
00115     void setBoardingDateOffsetList (const DateOffsetList_T& iList) {
00116         _boardingDateOffsetList = iList;
00117     }
00118
00120     void setNbOfAirlines (const stdair::NbOfAirlines_T& iNbOfAirlines) {
00121         _nbOfAirlines = iNbOfAirlines;
00122     }
00123
00125     void setElapsedTime (const stdair::Duration_T& iElapsed) {
00126         _elapsed = iElapsed;
00127     }
00128
00130     void setBoardingTime (const stdair::Duration_T& iBoardingTime) {
00131         _boardingTime = iBoardingTime;
00132     }
00133
00134
00135 public:
00136     // //////////// Business methods ////////////
00138     const bool isValid () const {
00139         return _period.isValid ();
00140     }
00141
00142
00143 public:
00144     // //////////// Display support methods ////////////
00150     void toStream (std::ostream& ioOut) const;
00151
00157     void fromStream (std::istream& ioIn);
00158
00168     const std::string toString() const;
00169
00170
00171 public:
00172     // //////////// (Boost) Serialisation support methods ////////////
00176     template<class Archive>
00177     void serialize (Archive& ar, const unsigned int iFileVersion);
00178
00179 private:
00184     void serialisationImplementationExport() const;
00185     void serialisationImplementationImport();
00186
00187
00188 private:
00189     // //////////// Attributes ////////////
00193     stdair::PeriodStruct _period;
00194
00198     stdair::Duration_T _boardingTime;
00199
00203     stdair::Duration_T _elapsed;
00204
00209     DateOffsetList_T _boardingDateOffsetList;
00210
00214     stdair::NbOfAirlines_T _nbOfAirlines;
00215 };
00216
```

```
00217 }  
00218 #endif // __AIRTSP_BOM_SEGMENTPATHPERIODKEY_HPP
```



## 26.77 airtsp/bom/SegmentPathPeriodTypes.hpp File Reference

```
#include <map>
#include <vector>
#include <list>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/key_types.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

### Typedefs

- typedef std::list< SegmentPathPeriod \* > [AIRTSP::SegmentPathPeriodList\\_T](#)
- typedef std::multimap< const stdair::MapKey\_T, SegmentPathPeriod \* > [AIRTSP::SegmentPathPeriodMultimap\\_T](#)
- typedef std::vector< const SegmentPathPeriod \* > [AIRTSP::SegmentPathPeriodLightList\\_T](#)
- typedef std::vector< SegmentPathPeriodLightList\_T > [AIRTSP::SegmentPathPeriodListList\\_T](#)
- typedef std::vector< stdair::DateOffset\_T > [AIRTSP::DateOffsetList\\_T](#)

## 26.78 SegmentPathPeriodTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __AIRTSP_BOM_SEGMENTPATHPERIODTYPES_HPP
00003 #define __AIRTSP_BOM_SEGMENTPATHPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <vector>
00011 #include <list>
00012 // StdAir
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_date_time_types.hpp>
00015 #include <stdair/bom/key_types.hpp>
00016
00017 namespace AIRTSP {
00018
00020     class SegmentPathPeriod;
00021
00023     typedef std::list<SegmentPathPeriod*> SegmentPathPeriodList_T;
00024
00026     typedef std::multimap<const stdair::MapKey_T,
00027                          SegmentPathPeriod*> SegmentPathPeriodMultimap_T;
00028
00030     typedef std::vector<const SegmentPathPeriod*> SegmentPathPeriodLightList_T;
00031     typedef std::vector<SegmentPathPeriodLightList_T>SegmentPathPeriodListList_T;
00032
00035     typedef std::vector<stdair::DateOffset_T> DateOffsetList_T;
00036
00037 }
00038 #endif // __AIRTSP_BOM_SEGMENTPATHPERIODTYPES_HPP
00039
```

## 26.79 airtsp/bom/SegmentPeriodHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <airtsp/bom/SegmentPeriodHelper.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.80 SegmentPeriodHelper.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasConst_General.hpp>
00008 #include <stdair/bom/SegmentPeriod.hpp>
00009 // AirTSP
00010 #include <airtsp/bom/SegmentPeriodHelper.hpp>
00011
00012 namespace AIRTSP {
00013 // //////////////////////////////////////
00014 void SegmentPeriodHelper::fill (stdair::SegmentPeriod& ioSegmentPeriod,
00015                                const SegmentStruct& iSegmentStruct) {
00016     // Browse the list of segment cabins and fill the cabin booking
00017     // class map of the BOM segment period.
00018     for (SegmentCabinStructList_T::const_iterator itCabin =
00019          iSegmentStruct._cabinList.begin();
00020          itCabin != iSegmentStruct._cabinList.end(); ++itCabin) {
00021         const SegmentCabinStruct& lSegmentCabinStruct = *itCabin;
00022         ioSegmentPeriod.
00023             addCabinBookingClassList (lSegmentCabinStruct._cabinCode,
00024                                     lSegmentCabinStruct._classes);
00025     }
00026 }
00027
00028 // //////////////////////////////////////
00029 void SegmentPeriodHelper::fill (stdair::SegmentPeriod& ioSegmentPeriod,
00030                                const LegStructList_T& iLegList) {
00031
00032     const stdair::AirportCode_T& lBoardingPoint =
00033         ioSegmentPeriod.getBoardingPoint ();
00034     const stdair::AirportCode_T& lOffPoint = ioSegmentPeriod.getOffPoint ();
00035     stdair::Duration_T lElapsedTime;
00036
00037     // Find the leg which has the same boarding point.
00038     LegStructList_T::const_iterator itLeg = iLegList.begin ();
00039     while (itLeg != iLegList.end()) {
00040         const LegStruct& lLeg = *itLeg;
00041         if (lLeg._boardingPoint == lBoardingPoint) {
00042             break;
00043         } else {
00044             ++itLeg;
00045         }
00046     }
00047     assert (itLeg != iLegList.end());
00048     const LegStruct& lFirstLeg = *itLeg;
00049     stdair::AirportCode_T lCurrentOffPoint = lFirstLeg._offPoint;
00050     stdair::Duration_T lCurrentOffTime = lFirstLeg._offTime;
00051
00052     // Update the elapsed time.
00053     lElapsedTime += lFirstLeg._elapsed;
00054
00055     // Find the last used leg.
00056     while (lCurrentOffPoint != lOffPoint) {
00057         ++itLeg;
00058         assert (itLeg != iLegList.end());
00059
00060         const LegStruct& lCurrentLeg = *itLeg;
00061         assert (lCurrentOffPoint == lCurrentLeg._boardingPoint);
00062         // As the boarding point of the current leg is the same as the off point
00063         // of the previous leg (by construction), there is no time difference.
00064         const stdair::Duration_T lStopOverTime =
00065             lCurrentLeg._boardingTime - lCurrentOffTime;

```

```
00066         lElapsedTime += lStopOverTime;
00067
00068         // Add the elapsed time of the current leg
00069         lElapsedTime += lCurrentLeg._elapsed;
00070
00071         lCurrentOffTime = lCurrentLeg._offTime;
00072         lCurrentOffPoint = lCurrentLeg._offPoint;
00073     }
00074     const LegStruct& lLastLeg = *itLeg;
00075
00076     // Update the attributes of the segment-period.
00077     ioSegmentPeriod.setBoardingTime (lFirstLeg._boardingTime);
00078     ioSegmentPeriod.setOffTime (lLastLeg._offTime);
00079     ioSegmentPeriod.setBoardingDateOffset (lFirstLeg._boardingDateOffset);
00080     ioSegmentPeriod.setOffDateOffset (lLastLeg._offDateOffset);
00081     ioSegmentPeriod.setElapsedTime (lElapsedTime);
00082 }
00083
00084 }
```

## 26.81 airtsp/bom/SegmentPeriodHelper.hpp File Reference

```
#include <airtsp/bom/LegStruct.hpp>
#include <airtsp/bom/SegmentStruct.hpp>
```

### Classes

- class [AIRTSP::SegmentPeriodHelper](#)  
*Class representing the actual business functions for an airline segment-period.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.82 SegmentPeriodHelper.hpp

```
00001 #ifndef __AIRTSP_BOM_SEGMENTPERIODHELPER_HPP
00002 #define __AIRTSP_BOM_SEGMENTPERIODHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // AirTSP
00008 #include <airtsp/bom/LegStruct.hpp>
00009 #include <airtsp/bom/SegmentStruct.hpp>
00010
00011 // Forward declarations
00012 namespace stdair {
00013     class SegmentPeriod;
00014 }
00015
00016 namespace AIRTSP {
00017
00022     class SegmentPeriodHelper {
00023     public:
00024         // ////////// Business Methods //////////
00029         static void fill (stdair::SegmentPeriod&, const SegmentStruct&);
00030
00035         static void fill (stdair::SegmentPeriod&, const LegStructList_T&);
00036     };
00037
00038 }
00039 #endif // __AIRTSP_BOM_SEGMENTPERIODHELPER_HPP
```

## 26.83 airtsp/bom/SegmentStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentDate.hpp>
#include <airtsp/bom/SegmentStruct.hpp>
```

### Namespaces

- namespace [AIRTSP](#)



## 26.84 SegmentStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/SegmentDate.hpp>
00009 // AirTSP
00010 #include <airtsp/bom/SegmentStruct.hpp>
00011
00012 namespace AIRTSP {
00013
00014 // //////////////////////////////////////
00015 const std::string SegmentStruct::describe() const {
00016     std::ostringstream ostr;
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     for (SegmentCabinStructList_T::const_iterator itCabin =
00025         _cabinList.begin(); itCabin != _cabinList.end(); itCabin++) {
00026         const SegmentCabinStruct& lCabin = *itCabin;
00027         ostr << lCabin.describe();
00028     }
00029     ostr << std::endl;
00030
00031     return ostr.str();
00032 }
00033
00034 // //////////////////////////////////////
00035 void SegmentStruct::fill (stdair::SegmentDate& ioSegmentDate) const {
00036     // Note that some parameters (boarding date, boarding time, off
00037     // date, off time, elapsed time) are set by
00038     // SegmentDate::fillFromRouting() when the routing (with legs) is
00039     // built. So, it is useless to set those parameters here.
00040
00041     // At that time, there are no other parameters.
00042 }
00043
00044 }

```

## 26.85 airtsp/bom/SegmentStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airtsp/bom/SegmentCabinStruct.hpp>
```

### Classes

- struct [AIRTSP::SegmentStruct](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

### Typedefs

- typedef std::vector< SegmentStruct > [AIRTSP::SegmentStructList\\_T](#)

## 26.86 SegmentStruct.hpp

```

00001 #ifndef __AIRTSP_BOM_SEGMENTSTRUCT_HPP
00002 #define __AIRTSP_BOM_SEGMENTSTRUCT_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/basic/StructAbstract.hpp>
00013 // AirTSP
00014 #include <airtsp/bom/SegmentCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentDate;
00019 }
00020
00021 namespace AIRTSP {
00022
00023     struct SegmentStruct : public stdair::StructAbstract {
00024         // Attributes
00025         stdair::AirportCode_T _boardingPoint;
00026         stdair::Date_T _boardingDate;
00027         stdair::Duration_T _boardingTime;
00028         stdair::AirportCode_T _offPoint;
00029         stdair::Date_T _offDate;
00030         stdair::Duration_T _offTime;
00031         stdair::Duration_T _elapsed;
00032         SegmentCabinStructList_T _cabinList;
00033
00034         void fill (stdair::SegmentDate&) const;
00035
00036         const std::string describe() const;
00037     };
00038
00039     typedef std::vector<SegmentStruct> SegmentStructList_T;
00040
00041 }
00042
00043 #endif // __AIRTSP_BOM_SEGMENTSTRUCT_HPP

```

## 26.87 airtsp/command/InventoryGenerator.cpp File Reference

```
#include <cassert>
#include <boost/date_time/date_iterator.hpp>
#include <stdair/stdair_basic_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/AirlineFeature.hpp>
#include <stdair/bom/FlightPeriod.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/bom/FlightPeriodStruct.hpp>
#include <airtsp/bom/SegmentPeriodHelper.hpp>
#include <airtsp/command/InventoryGenerator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.88 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_basic_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/AirlineFeature.hpp>
00015 #include <stdair/bom/FlightPeriod.hpp>
00016 #include <stdair/bom/SegmentPeriod.hpp>
00017 #include <stdair/factory/FacBomManager.hpp>
00018 #include <stdair/service/Logger.hpp>
00019 // AirTSP
00020 #include <airtsp/bom/FlightPeriodStruct.hpp>
00021 #include <airtsp/bom/SegmentPeriodHelper.hpp>
00022 #include <airtsp/command/InventoryGenerator.hpp>
00023
00024 namespace AIRTSP {
00025
00026 // //////////////////////////////////////
00027 void InventoryGenerator::
00028     createFlightPeriod (stdair::BomRoot& ioBomRoot,
00029                         const FlightPeriodStruct& iFlightPeriodStruct) {
00030
00031     const stdair::AirlineCode_T& lAirlineCode = iFlightPeriodStruct._airlineCode;
00032
00033     // Instantiate an inventory object (if not exist)
00034     // for the given key (airline code)
00035     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00036         getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00037     if (lInventory_ptr == NULL) {
00038         stdair::InventoryKey lKey (lAirlineCode);
00039
00040         lInventory_ptr =
00041             &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00042         stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00043         stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00044
00045         // Add the airline feature object to the inventory
00046         const stdair::AirlineFeatureKey lAirlineFeatureKey (lAirlineCode);
00047         stdair::AirlineFeature& lAirlineFeature =
00048             stdair::FacBom<stdair::AirlineFeature>::instance().create (lAirlineFeatur
00049 eKey);
00049         stdair::FacBomManager::setAirlineFeature (*lInventory_ptr,
00050                                                     lAirlineFeature);
00051         stdair::FacBomManager::linkWithParent (*lInventory_ptr, lAirlineFeature);
00052         // Link the airline feature object with the top of the BOM tree
00053         stdair::FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);
00054     }
00055     assert (lInventory_ptr != NULL);
00056
00057     // Create the flight-period key.
00058     const stdair::PeriodStruct lPeriod (iFlightPeriodStruct._dateRange,
00059                                         iFlightPeriodStruct._dow);
00060     const stdair::FlightPeriodKey
00061         lFlightPeriodKey (iFlightPeriodStruct._flightNumber, lPeriod);
00062
00063     // Check that the flight-period object is not already created.

```

```

00064     stdair::FlightPeriod* lFlightPeriod_ptr = stdair::BomManager::
00065         getObjectPtr<stdair::FlightPeriod> (*lInventory_ptr,
00066                                             lFlightPeriodKey.toString());
00067     if (lFlightPeriod_ptr != NULL) {
00068         throw stdair::ObjectCreationgDuplicationException ("");
00069     }
00070     assert (lFlightPeriod_ptr == NULL);
00071
00072     // Instantiate a flight-period object with the given key.
00073     lFlightPeriod_ptr = &stdair::FacBom<stdair::FlightPeriod>::
00074         instance().create (lFlightPeriodKey);
00075     stdair::FacBomManager::addToListAndMap (*lInventory_ptr, *lFlightPeriod_ptr);
00076
00077     stdair::FacBomManager::linkWithParent (*lInventory_ptr, *lFlightPeriod_ptr);
00078
00079     // Create the segment-periods.
00080     createSegmentPeriods (*lFlightPeriod_ptr, iFlightPeriodStruct);
00081 }
00082 // //////////////////////////////////////
00083 void InventoryGenerator::
00084 createSegmentPeriods (stdair::FlightPeriod& ioFlightPeriod,
00085                      const FlightPeriodStruct& iFlightPeriodStruct) {
00086
00087     // Iterate on the segment strutures.
00088     const SegmentStructList_T& lSegmentList = iFlightPeriodStruct._segmentList;
00089     for (SegmentStructList_T::const_iterator itSegment = lSegmentList.begin();
00090          itSegment != lSegmentList.end(); ++itSegment) {
00091
00092         const SegmentStruct& lSegment = *itSegment;
00093
00094         // Set the segment-period primary key.
00095         const stdair::AirportCode_T& lBoardingPoint = lSegment._boardingPoint;
00096         const stdair::AirportCode_T& lOffPoint = lSegment._offPoint;
00097         const stdair::SegmentPeriodKey lSegmentPeriodKey (lBoardingPoint,
00098                                                         lOffPoint);
00099
00100         // Instantiate a segment-perioed with the key.
00101         stdair::SegmentPeriod& lSegmentPeriod = stdair::
00102             FacBom<stdair::SegmentPeriod>::instance().create (lSegmentPeriodKey);
00103         stdair::FacBomManager::addToListAndMap (ioFlightPeriod, lSegmentPeriod);
00104         stdair::FacBomManager::linkWithParent (ioFlightPeriod, lSegmentPeriod);
00105
00106         // Set the segment-period attributes.
00107         SegmentPeriodHelper::fill (lSegmentPeriod, lSegment);
00108         SegmentPeriodHelper::fill (lSegmentPeriod, iFlightPeriodStruct._legList);
00109     }
00110 }
00111
00112 }

```

## 26.89 airtsp/command/InventoryGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
```

### Classes

- class [AIRTSP::InventoryGenerator](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)
- namespace [AIRTSP::ScheduleParserHelper](#)

**26.90 InventoryGenerator.hpp**

```

00001 #ifndef __AIRTSP_CMD_INVENTORYGENERATOR_HPP
00002 #define __AIRTSP_CMD_INVENTORYGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirTSP
00010 #include <airtsp/AIRTSP_Types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     class BomRoot;
00015     class FlightPeriod;
00016 }
00017
00018 namespace AIRTSP {
00019
00020     // Forward declarations
00021     struct FlightPeriodStruct;
00022     struct LegStruct;
00023     struct SegmentStruct;
00024     struct LegCabinStruct;
00025     struct SegmentCabinStruct;
00026     namespace ScheduleParserHelper {
00027         struct doEndFlight;
00028     }
00029
00030     class InventoryGenerator : public stdair::CmdAbstract {
00031     // Only the following class may use methods of InventoryGenerator.
00032     // Indeed, as those methods build the BOM, it is not good to expose
00033     // them publicly.
00034     friend class FlightPeriodFileParser;
00035     friend class FFFlightPeriodFileParser;
00036     friend struct ScheduleParserHelper::doEndFlight;
00037     friend class ScheduleParser;
00038
00039     private:
00040         static void createFlightPeriod (stdair::BomRoot&,
00041                                         const FlightPeriodStruct&);
00042
00043         static void createSegmentPeriods (stdair::FlightPeriod&,
00044                                           const FlightPeriodStruct&);
00045
00046     };
00047
00048 }
00049
00050 #endif // __AIRTSP_CMD_INVENTORYGENERATOR_HPP

```



## 26.91 airtsp/command/OnDParser.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <airtsp/command/OnDParserHelper.hpp>
#include <airtsp/command/OnDParser.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

## 26.92 OnDParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasFileMgr.hpp>
00008 #include <stdair/service/Logger.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 // AirTSP
00011 #include <airtsp/command/OnDParserHelper.hpp>
00012 #include <airtsp/command/OnDParser.hpp>
00013
00014 namespace AIRTSP {
00015
00016 // //////////////////////////////////////
00017 void OnDParser::generateOnDPeriods (const stdair::ODFilePath& iODFilename,
00018                                     stdair::BomRoot& ioBomRoot) {
00019
00020     const stdair::Filename_T lFilename = iODFilename.name();
00021
00022     // Check that the file path given as input corresponds to an actual file
00023     const bool doesExistAndIsReadable =
00024         stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00025
00026     if (doesExistAndIsReadable == false) {
00027         STDAIR_LOG_ERROR ("The O&D input file, '" << lFilename
00028                             << "', can not be retrieved on the file-system");
00029         throw ONDInputFileNotFoundException ("The O&D file " + lFilename
00030                                             + " does not exist or can not be "
00031                                             + "read");
00032     }
00033
00034     // Initialise the O&D-Period file parser.
00035     ONDPeriodFileParser lOnDPeriodParser (lFilename, ioBomRoot);
00036
00037     // Parse the CSV-formatted O&D input file, and generate the
00038     // corresponding O&D-Period for the airlines.
00039     lOnDPeriodParser.generateOnDPeriods();
00040 }
00041
00042 }

```

## 26.93 airtsp/command/OnDParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

### Classes

- class [AIRTSP::OnDParser](#)  
*Class wrapping the parser entry point.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.94 OnDParser.hpp

```
00001 #ifndef __AIRTSP_CMD_ONDPARSER_HPP
00002 #define __AIRTSP_CMD_ONDPARSER_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_file.hpp>
00012 #include <stdair/command/CmdAbstract.hpp>
00013
00015 namespace stdair {
00016     class BomRoot;
00017 }
00018
00019 namespace AIRTSP {
00020
00024     class OnDParser : public stdair::CmdAbstract {
00025     public:
00032         static void generateOnDPeriods (const stdair::ODFilePath&,
00033                                         stdair::BomRoot&);
00034     };
00035
00036 }
00037 #endif // __AIRTSP_CMD_ONDPARSER_HPP
```

## 26.95 airtsp/command/OnDParserHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/command/OnDParserHelper.hpp>
#include <airtsp/command/OnDPeriodGenerator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)
- namespace [AIRTSP::OnDParserHelper](#)

### Functions

- [chset\\_t AIRTSP::OnDParserHelper::alpha\\_cap\\_set\\_p](#) ("A-Z")
- [repeat\\_p\\_t AIRTSP::OnDParserHelper::airport\\_p](#) ([chset\\_t](#)("0-9A-Z").derived(), 3, 3)
- [repeat\\_p\\_t AIRTSP::OnDParserHelper::airline\\_code\\_p](#) ([alpha\\_cap\\_set\\_p](#).derived(), 2, 3)
- [bounded4\\_p\\_t AIRTSP::OnDParserHelper::year\\_p](#) ([uint4\\_p](#).derived(), 2000u, 2099u)
- [bounded2\\_p\\_t AIRTSP::OnDParserHelper::month\\_p](#) ([uint2\\_p](#).derived(), 1u, 12u)
- [bounded2\\_p\\_t AIRTSP::OnDParserHelper::day\\_p](#) ([uint2\\_p](#).derived(), 1u, 31u)
- [bounded2\\_p\\_t AIRTSP::OnDParserHelper::hours\\_p](#) ([uint2\\_p](#).derived(), 0u, 23u)
- [bounded2\\_p\\_t AIRTSP::OnDParserHelper::minutes\\_p](#) ([uint2\\_p](#).derived(), 0u, 59u)
- [bounded2\\_p\\_t AIRTSP::OnDParserHelper::seconds\\_p](#) ([uint2\\_p](#).derived(), 0u, 59u)
- [chset\\_t AIRTSP::OnDParserHelper::class\\_code\\_p](#) ("A-Z")

### Variables

- [uint2\\_p\\_t AIRTSP::OnDParserHelper::uint2\\_p](#)
- [uint4\\_p\\_t AIRTSP::OnDParserHelper::uint4\\_p](#)
- [uint1\\_4\\_p\\_t AIRTSP::OnDParserHelper::uint1\\_4\\_p](#)

## 26.96 OnDParserHelper.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasFileMgr.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AirTSP
00011 #include <airtsp/command/OnDParserHelper.hpp>
00012 #include <airtsp/command/OnDPeriodGenerator.hpp>
00013
00014 namespace AIRTSP {
00015
00016     namespace OnDParserHelper {
00017
00018         // //////////////////////////////////////
00019         //
00020         // Semantic actions
00021         //
00022         // //////////////////////////////////////
00023
00024         ParserSemanticAction::
00025         ParserSemanticAction (OnDPeriodStruct& ioOnDPeriod)
00026             : _onDPeriod (ioOnDPeriod) {
00027         }
00028
00029         // //////////////////////////////////////
00030         storeOrigin::storeOrigin (OnDPeriodStruct& ioOnDPeriod)
00031             : ParserSemanticAction (ioOnDPeriod) {
00032         }
00033
00034         // //////////////////////////////////////
00035         void storeOrigin::operator() (iterator_t iStr,
00036                                     iterator_t iStrEnd) const {
00037             std::string lOrigin (iStr, iStrEnd);
00038             //STDAIR_LOG_DEBUG ( "Origin: " << lOrigin << std::endl);
00039
00040             // Set the origin
00041             _onDPeriod._origin = lOrigin;
00042             _onDPeriod._nbOfAirlines = 0;
00043             _onDPeriod._airlineCode = "";
00044             _onDPeriod._classCode = "";
00045             _onDPeriod._airlineCodeList.clear();
00046             _onDPeriod._classCodeList.clear();
00047         }
00048
00049         // //////////////////////////////////////
00050         storeDestination::storeDestination (OnDPeriodStruct& ioOnDPeriod)
00051             : ParserSemanticAction (ioOnDPeriod) {
00052         }
00053
00054         // //////////////////////////////////////
00055         void storeDestination::operator() (iterator_t iStr,
00056                                     iterator_t iStrEnd) const {
00057             std::string lDestination (iStr, iStrEnd);
00058             //STDAIR_LOG_DEBUG ("Destination: " << lDestination << std::endl);
00059
00060             // Set the destination
00061             _onDPeriod._destination = lDestination;
00062         }
00063
00064         // //////////////////////////////////////
00065         storeDateRangeStart::

```

```

00066     storeDateRangeStart (OnDPeriodStruct& ioOnDPeriod)
00067     : ParserSemanticAction (ioOnDPeriod) {
00068     }
00069
00070     // //////////////////////////////////////
00071 void storeDateRangeStart::operator() (iterator_t iStr,
00072                                     iterator_t iStrEnd) const {
00073     _onDPeriod._dateRangeStart = _onDPeriod.getDate();
00074     /*STDAIR_LOG_DEBUG ("Date Range Start: "
00075     << _onDPeriod._dateRangeStart << std::endl);*/
00076
00077     // Reset the number of seconds
00078     _onDPeriod._itSeconds = 0;
00079 }
00080
00081     // //////////////////////////////////////
00082 storeDateRangeEnd::
00083 storeDateRangeEnd (OnDPeriodStruct& ioOnDPeriod)
00084     : ParserSemanticAction (ioOnDPeriod) {
00085     }
00086
00087     // //////////////////////////////////////
00088 void storeDateRangeEnd::operator() (iterator_t iStr,
00089                                     iterator_t iStrEnd) const {
00090     // As a Boost date period (COM::DatePeriod_T) defines the last day of
00091     // the period to be end-date - one day, we have to add one day to that
00092     // end date before.
00093     const stdair::DateOffset_T oneDay (1);
00094     _onDPeriod._dateRangeEnd = _onDPeriod.getDate() + oneDay;
00095     /*STDAIR_LOG_DEBUG ( "Date Range End: "
00096     << _onDPeriod._dateRangeEnd << std::endl);*/
00097
00098     // Transform the date pair (i.e., the date range) into a date period
00099     _onDPeriod._datePeriod =
00100     stdair::DatePeriod_T (_onDPeriod._dateRangeStart,
00101     _onDPeriod._dateRangeEnd);
00102
00103     // Reset the number of seconds
00104     _onDPeriod._itSeconds = 0;
00105 }
00106
00107     // //////////////////////////////////////
00108 storeStartRangeTime::
00109 storeStartRangeTime (OnDPeriodStruct& ioOnDPeriod)
00110     : ParserSemanticAction (ioOnDPeriod) {
00111     }
00112
00113     // //////////////////////////////////////
00114 void storeStartRangeTime::operator() (iterator_t iStr,
00115                                     iterator_t iStrEnd) const {
00116     _onDPeriod._timeRangeStart = _onDPeriod.getTime();
00117
00118     // Reset the number of seconds
00119     _onDPeriod._itSeconds = 0;
00120 }
00121
00122     // //////////////////////////////////////
00123 storeEndRangeTime::
00124 storeEndRangeTime (OnDPeriodStruct& ioOnDPeriod)
00125     : ParserSemanticAction (ioOnDPeriod) {
00126     }
00127
00128     // //////////////////////////////////////
00129 void storeEndRangeTime::operator() (iterator_t iStr,
00130                                     iterator_t iStrEnd) const {
00131     _onDPeriod._timeRangeEnd = _onDPeriod.getTime();
00132

```

```

00133         // Reset the number of seconds
00134         _onDPeriod._itSeconds = 0;
00135     }
00136
00137     // //////////////////////////////////////
00138     storeAirlineCode::
00139     storeAirlineCode (OnDPeriodStruct& ioOnDPeriod)
00140         : ParserSemanticAction (ioOnDPeriod) {
00141     }
00142
00143     // //////////////////////////////////////
00144     void storeAirlineCode::operator() (iterator_t iStr,
00145                                       iterator_t iStrEnd) const {
00146         const std::string lAirlineCodeStr (iStr, iStrEnd);
00147         const stdair::AirlineCode_T lAirlineCode(lAirlineCodeStr);
00148         // Test if the OnD Period Struct stands for interline products
00149         if (_onDPeriod._airlineCodeList.size() > 0) {
00150             // update the airline code
00151             std::ostringstream ostr;
00152             ostr << _onDPeriod._airlineCode << lAirlineCode;
00153             _onDPeriod._airlineCode = ostr.str();
00154             // Update the number of airlines if necessary
00155             const stdair::AirlineCode_T lPreviousAirlineCode =
00156                 _onDPeriod._airlineCodeList.back();
00157             if (lPreviousAirlineCode != lAirlineCode) {
00158                 _onDPeriod._nbOfAirlines = _onDPeriod._nbOfAirlines + 1;
00159             }
00160         }
00161         else {
00162             _onDPeriod._airlineCode = lAirlineCode;
00163             _onDPeriod._nbOfAirlines = 1;
00164         }
00165         _onDPeriod._airlineCodeList.push_back (lAirlineCode);
00166
00167         //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode << std::endl);
00168     }
00169
00170     // //////////////////////////////////////
00171     storeClassCode::
00172     storeClassCode (OnDPeriodStruct& ioOnDPeriod)
00173         : ParserSemanticAction (ioOnDPeriod) {
00174     }
00175
00176     // //////////////////////////////////////
00177     void storeClassCode::operator() (char iChar) const {
00178         std::ostringstream ostr;
00179         ostr << iChar;
00180         std::string classCodeStr = ostr.str();
00181         const stdair::ClassCode_T lClassCode (classCodeStr);
00182         _onDPeriod._classCodeList.push_back (lClassCode);
00183         /*STDAIR_LOG_DEBUG ("Class Code: "
00184                            << lClassCode << std::endl);*/
00185         // Insertion of this class Code in the whole classCode name
00186         std::ostringstream ostrr;
00187         ostrr << _onDPeriod._classCode << classCodeStr;
00188         _onDPeriod._classCode = ostrr.str();
00189     }
00190
00191
00192     // //////////////////////////////////////
00193     doEndOnD::doEndOnD (stdair::BomRoot& ioBomRoot, OnDPeriodStruct& ioOnDPeriod)
00194         : ParserSemanticAction (ioOnDPeriod),
00195         _bomRoot (ioBomRoot) {
00196     }
00197
00198     // //////////////////////////////////////

```



```

00199     void doEndOnD::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00200
00201         // DEBUG: Display the result
00202         // STDAIR_LOG_DEBUG ("FareRule " << _onDPeriod.describe());
00203
00204         // Generation of the O&D-Period object.
00205         OnDPeriodGenerator::createOnDPeriod (_bomRoot, _onDPeriod);
00206     }
00207
00208     // //////////////////////////////////////
00209     //
00210     // Utility Parsers
00211     //
00212     // //////////////////////////////////////
00213
00215     uint2_p_t uint2_p;
00216
00218     uint4_p_t uint4_p;
00219
00221     uint1_4_p_t uint1_4_p;
00222
00224     chset_t alpha_cap_set_p ("A-Z");
00225
00227     repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3);
00228
00230     repeat_p_t airline_code_p (alpha_cap_set_p.derived(), 2, 3);
00231
00233     bounded4_p_t year_p (uint4_p.derived(), 2000u, 2099u);
00234
00236     bounded2_p_t month_p (uint2_p.derived(), 1u, 12u);
00237
00239     bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00240
00242     bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u);
00243
00245     bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u);
00246
00248     bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u);
00249
00251     chset_t class_code_p ("A-Z");
00252
00254     //
00255     // (Boost Spirit) Grammar Definition
00256     //
00258
00259     // //////////////////////////////////////
00260     OnDParser::
00261     OnDParser (stdair::BomRoot& ioBomRoot, OnDPeriodStruct& ioOnDPeriod)
00262         : _bomRoot (ioBomRoot), _onDPeriod (ioOnDPeriod) {
00263     }
00264
00265     // //////////////////////////////////////
00266     template<typename ScannerT>
00267     OnDParser::definition<ScannerT>::definition (OnDParser const& self) {
00268
00269         ond_list = *( boost::spirit::classic::comment_p("//")
00270                     | boost::spirit::classic::comment_p("/*", "*/")
00271                     | ond )
00272
00273         ;
00274
00275         ond = ond_key
00276         >> +( ';' >> segment )
00277         >> ond_end[doEndOnD(self._bomRoot, self._onDPeriod)]
00278
00279         ;
00280
00281         ond_end = boost::spirit::classic::ch_p(';')
00282
00283         ;

```

```

00281
00282     ond_key = (airport_p) [storeOrigin(self._onDPeriod)]
00283     >> ';' >> (airport_p) [storeDestination(self._onDPeriod)]
00284     >> ';' >> date[storeDateRangeStart(self._onDPeriod)]
00285     >> ';' >> date[storeDateRangeEnd(self._onDPeriod)]
00286     >> ';' >> time[storeStartRangeTime(self._onDPeriod)]
00287     >> ';' >> time[storeEndRangeTime(self._onDPeriod)]
00288     ;
00289
00290     date = boost::spirit::classic::
00291         lexeme_d[(year_p) [boost::spirit::classic::
00292             assign_a(self._onDPeriod._itYear)]
00293             >> '-'
00294             >> (month_p) [boost::spirit::classic::
00295                 assign_a(self._onDPeriod._itMonth)]
00296             >> '-'
00297             >> (day_p) [boost::spirit::classic::
00298                 assign_a(self._onDPeriod._itDay)]]
00299     ;
00300
00301     time = boost::spirit::classic::
00302         lexeme_d[(hours_p) [boost::spirit::classic::
00303             assign_a(self._onDPeriod._itHours)]
00304             >> ':'
00305             >> (minutes_p) [boost::spirit::classic::
00306                 assign_a(self._onDPeriod._itMinutes)]
00307             >> !(':' >> (seconds_p) [boost::spirit::classic::
00308                 assign_a(self._onDPeriod._itSeconds)])]
00309     ;
00310
00311     segment = boost::spirit::classic::
00312         lexeme_d[(airline_code_p) [storeAirlineCode(self._onDPeriod)]]
00313         >> ';' >> (class_code_p) [storeClassCode(self._onDPeriod)]
00314     ;
00315
00316     //BOOST_SPIRIT_DEBUG_NODE (OnDParser);
00317     BOOST_SPIRIT_DEBUG_NODE (ond_list);
00318     BOOST_SPIRIT_DEBUG_NODE (ond);
00319     BOOST_SPIRIT_DEBUG_NODE (segment);
00320     BOOST_SPIRIT_DEBUG_NODE (ond_key);
00321     BOOST_SPIRIT_DEBUG_NODE (ond_end);
00322     BOOST_SPIRIT_DEBUG_NODE (date);
00323     BOOST_SPIRIT_DEBUG_NODE (time);
00324 }
00325
00326
00327 // //////////////////////////////////////
00328 template<typename ScannerT>
00329 boost::spirit::classic::rule<ScannerT> const&
00330 OnDParser::definition<ScannerT>::start() const {
00331     return ond_list;
00332 }
00333 }
00334
00335 //
00336 // Entry class for the file parser
00337 //
00338
00339 // //////////////////////////////////////
00340 OnDPeriodFileParser::OnDPeriodFileParser (const stdair::Filename_T& iFilename,
00341     stdair::BomRoot& ioBomRoot)
00342 : _filename (iFilename), _bomRoot (ioBomRoot) {
00343     init();
00344 }
00345
00346
00347 // //////////////////////////////////////
00348 void OnDPeriodFileParser::init() {

```

```

00350     // Check that the file exists and is readable
00351     const bool doesExistAndIsReadable =
00352         stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00353
00354     if (doesExistAndIsReadable == false) {
00355         STDAIR_LOG_ERROR ("The O&D file " << _filename
00356             << " does not exist or can not be read.");
00357
00358         throw OnDInputFileNotFoundException ("The O&D file " + _filename
00359             + " does not exist or can not be read"
00360         );
00361     }
00362
00363     // Open the file
00364     _startIterator = iterator_t (_filename);
00365
00366     // Check that the filename exists and can be open
00367     if (!_startIterator) {
00368         STDAIR_LOG_DEBUG ("The O&D file " << _filename << " can not be open."
00369             << std::endl);
00370         throw OnDInputFileNotFoundException ("The file " + _filename
00371             + " does not exist or can not be read"
00372         );
00373     }
00374
00375     // Create an EOF iterator
00376     _endIterator = _startIterator.make_end();
00377
00378     // ////////////////////////////////////////
00379     bool OnDPeriodFileParser::generateOnDPeriods () {
00380         bool oResult = false;
00381
00382         STDAIR_LOG_DEBUG ("Parsing O&D input file: " << _filename);
00383
00384         // Initialise the parser (grammar) with the helper/staging structure.
00385         OnDParserHelper::OnDParser lodParser (_bomRoot, _onDPeriod);
00386
00387         // Launch the parsing of the file and, thanks to the doEndOnD
00388         // call-back structure, filling the worldSchedule (Fares)
00389         boost::spirit::classic::parse_info<iterator_t> info =
00390             boost::spirit::classic::parse (_startIterator, _endIterator, lodParser,
00391                 boost::spirit::classic::space_p);
00392
00393         // Retrieves whether or not the parsing was successful
00394         oResult = info.hit;
00395
00396         const std::string hasBeenFullyReadStr = (info.full == true) ? "": "not ";
00397         if (oResult == true) {
00398             STDAIR_LOG_DEBUG ("Parsing of O&D input file: " << _filename
00399                 << " succeeded: read " << info.length
00400                 << " characters. The input file has "
00401                 << hasBeenFullyReadStr
00402                 << "been fully read. Stop point: " << info.stop);
00403         } else {
00404             // TODO: decide whether to throw an exception
00405             STDAIR_LOG_ERROR ("Parsing of O&D input file: " << _filename
00406                 << " failed: read " << info.length
00407                 << " characters. The input file has "
00408                 << hasBeenFullyReadStr
00409                 << "been fully read. Stop point: " << info.stop);
00410         }
00411
00412         return oResult;
00413     }
00414 }

```

## 26.97 airtsp/command/OnDParserHelper.hpp File Reference

```
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
#include <airtsp/basic/BasParserTypes.hpp>
#include <airtsp/bom/OnDPeriodStruct.hpp>
```

### Classes

- struct [AIRTSP::OnDParserHelper::ParserSemanticAction](#)
- struct [AIRTSP::OnDParserHelper::storeOrigin](#)
- struct [AIRTSP::OnDParserHelper::storeDestination](#)
- struct [AIRTSP::OnDParserHelper::storeDateRangeStart](#)
- struct [AIRTSP::OnDParserHelper::storeDateRangeEnd](#)
- struct [AIRTSP::OnDParserHelper::storeStartRangeTime](#)
- struct [AIRTSP::OnDParserHelper::storeEndRangeTime](#)
- struct [AIRTSP::OnDParserHelper::storeAirlineCode](#)
- struct [AIRTSP::OnDParserHelper::storeClassCode](#)
- struct [AIRTSP::OnDParserHelper::doEndOnD](#)
- struct [AIRTSP::OnDParserHelper::OnDParser](#)
- struct [AIRTSP::OnDParserHelper::OnDParser::definition< ScannerT >](#)
- class [AIRTSP::OnDPeriodFileParser](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)
- namespace [AIRTSP::OnDParserHelper](#)

**26.98 OnDParserHelper.hpp**

```

00001 #ifndef __AIRTSP_CMD_ONDPARSERHELPER_HPP
00002 #define __AIRTSP_CMD_ONDPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost (Extended STL)
00010 #include <boost/date_time/posix_time/posix_time.hpp>
00011 #include <boost/date_time/gregorian/gregorian.hpp>
00012 // StdAir
00013 #include <stdair/command/CmdAbstract.hpp>
00014 // AirTSP
00015 #include <airtsp/AIRTSP_Types.hpp>
00016 #include <airtsp/basic/BasParserTypes.hpp>
00017 #include <airtsp/bom/OnDPeriodStruct.hpp>
00018
00019 // Forward declarations
00020 namespace stdair {
00021     class BomRoot;
00022 }
00023
00024 namespace AIRTSP {
00025
00026     namespace OnDParserHelper {
00027
00028         // //////////////////////////////////////
00029         //
00030         // Semantic actions
00031         //
00032         // //////////////////////////////////////
00033         struct ParserSemanticAction {
00034             ParserSemanticAction (OnDPeriodStruct&);
00035             OnDPeriodStruct& _onDPeriod;
00036         };
00037
00038         struct storeOrigin : public ParserSemanticAction {
00039             storeOrigin (OnDPeriodStruct&);
00040             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00041         };
00042
00043         struct storeDestination : public ParserSemanticAction {
00044             storeDestination (OnDPeriodStruct&);
00045             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00046         };
00047
00048         struct storeDateRangeStart : public ParserSemanticAction {
00049             storeDateRangeStart (OnDPeriodStruct&);
00050             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00051         };
00052
00053         struct storeDateRangeEnd : public ParserSemanticAction {
00054             storeDateRangeEnd (OnDPeriodStruct&);
00055             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00056         };
00057
00058         struct storeStartRangeTime : public ParserSemanticAction {
00059             storeStartRangeTime (OnDPeriodStruct&);
00060             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00061         };
00062
00063         struct storeEndRangeTime : public ParserSemanticAction {
00064             storeEndRangeTime (OnDPeriodStruct&);
00065             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00066         };
00067
00068         struct storeStartRangeTime : public ParserSemanticAction {
00069             storeStartRangeTime (OnDPeriodStruct&);
00070             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00071         };
00072
00073         struct storeEndRangeTime : public ParserSemanticAction {
00074             storeEndRangeTime (OnDPeriodStruct&);
00075             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00076         };
00077
00078         struct storeStartRangeTime : public ParserSemanticAction {
00079             storeStartRangeTime (OnDPeriodStruct&);
00080             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00081         };
00082
00083         struct storeEndRangeTime : public ParserSemanticAction {
00084             storeEndRangeTime (OnDPeriodStruct&);
00085             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00086         };
00087
00088     }
00089
00090 }
00091
00092 #endif

```

```

00087     };
00088
00090     struct storeAirlineCode : public ParserSemanticAction {
00092         storeAirlineCode (OnDPeriodStruct&);
00094         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00095     };
00096
00098     struct storeClassCode : public ParserSemanticAction {
00100         storeClassCode (OnDPeriodStruct&);
00102         void operator() (char iChar) const;
00103     };
00104
00106     struct doEndOnD : public ParserSemanticAction {
00108         doEndOnD (stdair::BomRoot&, OnDPeriodStruct&);
00110         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00112         stdair::BomRoot& _bomRoot;
00113     };
00114
00116     //
00117     // (Boost Spirit) Grammar Definition
00118     //
00120
00127     struct OnDParser :
00128         public boost::spirit::classic::grammar<OnDParser> {
00129
00130         OnDParser (stdair::BomRoot&, OnDPeriodStruct&);
00131
00132         template <typename ScannerT>
00133         struct definition {
00134             definition (OnDParser const& self);
00135
00136             // Instantiation of rules
00137             boost::spirit::classic::rule<ScannerT> ond_list, ond, segment,
00138                 ond_key, ond_end, date, time;
00139
00141             boost::spirit::classic::rule<ScannerT> const& start() const;
00142         };
00143
00144         // Parser Context
00145         stdair::BomRoot& _bomRoot;
00146         OnDPeriodStruct& _onDPeriod;
00147     };
00148 }
00149
00151 //
00152 // Entry class for the file parser
00153 //
00155
00161 class OnDPeriodFileParser : public stdair::CmdAbstract {
00162 public:
00164     OnDPeriodFileParser (const stdair::Filename_T& iFilename,
00165                         stdair::BomRoot& ioBomRoot);
00166
00168     bool generateOnDPeriods ();
00169
00170 private:
00172     void init();
00173
00174 private:
00175     // Attributes
00177     stdair::Filename_T _filename;
00178
00180     iterator_t _startIterator;
00181
00183     iterator_t _endIterator;
00184
00186     stdair::BomRoot& _bomRoot;

```

```
00187
00189     OnDPeriodStruct _onDPeriod;
00190 };
00191
00192 }
00193 #endif // __AIRTSP_CMD_ONDPARSERHELPER_HPP
```

## 26.99 airtsp/command/OnDPeriodGenerator.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/bom/OnDPeriodStruct.hpp>
#include <airtsp/command/OnDPeriodGenerator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)



**26.100 OnDPeriodGenerator.cpp**

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_date_time_types.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/factory/FacBomManager.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // AirTSP
00013 #include <airtsp/bom/OnDPeriodStruct.hpp>
00014 #include <airtsp/command/OnDPeriodGenerator.hpp>
00015
00016 namespace AIRTSP {
00017
00018 // //////////////////////////////////////
00019 void OnDPeriodGenerator::
00020 createOnDPeriod (stdair::BomRoot& ioBomRoot,
00021                 const OnDPeriodStruct& iOnDPeriodStruct) {
00022 }
00023 }
```

## 26.101 airtsp/command/OnDPeriodGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
```

### Classes

- class [AIRTSP::OnDPeriodGenerator](#)  
*Class handling the generation / instantiation of the O&D-Period BOM.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)
- namespace [AIRTSP::OnDParserHelper](#)

## 26.102 OnDPeriodGenerator.hpp

```
00001 #ifndef __AIRTSP_CMD_ONDPERIODGENERATOR_HPP
00002 #define __AIRTSP_CMD_ONDPERIODGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirTSP
00010 #include <airtsp/AIRTSP_Types.hpp>
00011
00012 namespace stdair {
00013     class BomRoot;
00014 }
00015
00016 namespace AIRTSP {
00017
00018     struct OnDPeriodStruct_T;
00019     namespace OnDParserHelper {
00020         struct doEndOnD;
00021     }
00022
00023     class OnDPeriodGenerator : public stdair::CmdAbstract {
00024     friend class OnDPeriodFileParser;
00025     friend struct OnDParserHelper::doEndOnD;
00026     friend class OnDParser;
00027
00028     private:
00029         static void createOnDPeriod (stdair::BomRoot&, const OnDPeriodStruct&);
00030     };
00031 }
00032 #endif // __AIRTSP_CMD_ONDPERIODGENERATOR_HPP
```

## 26.103 airtsp/command/ScheduleParser.cpp File Reference

```
#include <cassert>
#include <string>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/command/SegmentPathGenerator.hpp>
#include <airtsp/command/ScheduleParserHelper.hpp>
#include <airtsp/command/ScheduleParser.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.104 ScheduleParser.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirTSP
00012 #include <airtsp/command/SegmentPathGenerator.hpp>
00013 #include <airtsp/command/ScheduleParserHelper.hpp>
00014 #include <airtsp/command/ScheduleParser.hpp>
00015
00016 namespace AIRTSP {
00017
00018 // //////////////////////////////////////
00019 void ScheduleParser::generateInventories
00020 (const stdair::ScheduleFilePath& iScheduleFilename,
00021  stdair::BomRoot& ioBomRoot) {
00022
00023     const stdair::Filename_T lFilename = iScheduleFilename.name();
00024
00025     // Check that the file path given as input corresponds to an actual file
00026     const bool doesExistAndIsReadable =
00027         stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00028
00029     if (doesExistAndIsReadable == false) {
00030         STDAIR_LOG_ERROR ("The schedule input file, '" << lFilename
00031             << "'", can not be retrieved on the file-system");
00032         throw ScheduleInputFileNotFoundException ("The schedule file " + lFilename
00033             + " does not exist or can not "
00034             "be read");
00035     }
00036
00037     // Initialise the Flight-Period file parser.
00038     FlightPeriodFileParser lFlightPeriodParser (ioBomRoot, lFilename);
00039
00040     // Parse the CSV-formatted schedule input file, and generate the
00041     // corresponding Inventories for the airlines.
00042     lFlightPeriodParser.generateInventories();
00043 }
00044
00045 }

```

## 26.105 airtsp/command/ScheduleParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <stdair/stdair_file.hpp>
```

### Classes

- class [AIRTSP::ScheduleParser](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.106 ScheduleParser.hpp

```
00001 #ifndef __AIRTSP_CMD_SCHEDULEPARSER_HPP
00002 #define __AIRTSP_CMD_SCHEDULEPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012 #include <stdair/stdair_file.hpp>
00013
00014 // Forward declarations.
00015 namespace stdair {
00016     class BomRoot;
00017 }
00018
00019 namespace AIRTSP {
00020
00022     class ScheduleParser : public stdair::CmdAbstract {
00023     public:
00029         static void generateInventories (const stdair::ScheduleFilePath&,
00030                                         stdair::BomRoot&);
00031     };
00032 }
00033 #endif // __AIRTSP_CMD_SCHEDULEPARSER_HPP
```

## 26.107 airtsp/command/ScheduleParserHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/command/ScheduleParserHelper.hpp>
#include <airtsp/command/InventoryGenerator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)
- namespace [AIRTSP::ScheduleParserHelper](#)

### Functions

- repeat\_p\_t [AIRTSP::ScheduleParserHelper::airline\\_code\\_p](#) (chset\_t("0-9A-Z").derived(), 2, 3)
- bounded1\_4\_p\_t [AIRTSP::ScheduleParserHelper::flight\\_number\\_p](#) (uint1\_4\_p.derived(), 0u, 9999u)
- bounded4\_p\_t [AIRTSP::ScheduleParserHelper::year\\_p](#) (uint4\_p.derived(), 2000u, 2099u)
- bounded2\_p\_t [AIRTSP::ScheduleParserHelper::month\\_p](#) (uint2\_p.derived(), 1u, 12u)
- bounded2\_p\_t [AIRTSP::ScheduleParserHelper::day\\_p](#) (uint2\_p.derived(), 1u, 31u)
- repeat\_p\_t [AIRTSP::ScheduleParserHelper::dow\\_p](#) (chset\_t("0-1").derived().derived(), 7, 7)
- repeat\_p\_t [AIRTSP::ScheduleParserHelper::airport\\_p](#) (chset\_t("0-9A-Z").derived(), 3, 3)
- bounded2\_p\_t [AIRTSP::ScheduleParserHelper::hours\\_p](#) (uint2\_p.derived(), 0u, 23u)
- bounded2\_p\_t [AIRTSP::ScheduleParserHelper::minutes\\_p](#) (uint2\_p.derived(), 0u, 59u)
- bounded2\_p\_t [AIRTSP::ScheduleParserHelper::seconds\\_p](#) (uint2\_p.derived(), 0u, 59u)
- chset\_t [AIRTSP::ScheduleParserHelper::cabin\\_code\\_p](#) ("A-Z")
- repeat\_p\_t [AIRTSP::ScheduleParserHelper::key\\_p](#) (chset\_t("0-9A-Z").derived(), 1, 10)
- repeat\_p\_t [AIRTSP::ScheduleParserHelper::class\\_code\\_list\\_p](#) (chset\_t("A-Z").derived(), 1, 26)

### Variables

- int1\_p\_t [AIRTSP::ScheduleParserHelper::int1\\_p](#)
- uint2\_p\_t [AIRTSP::ScheduleParserHelper::uint2\\_p](#)
- uint4\_p\_t [AIRTSP::ScheduleParserHelper::uint4\\_p](#)
- uint1\_4\_p\_t [AIRTSP::ScheduleParserHelper::uint1\\_4\\_p](#)
- int1\_p\_t [AIRTSP::ScheduleParserHelper::family\\_code\\_p](#)



**26.108 ScheduleParserHelper.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasFileMgr.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AirTSP
00011 // #define BOOST_SPIRIT_DEBUG
00012 #include <airtsp/command/ScheduleParserHelper.hpp>
00013 #include <airtsp/command/InventoryGenerator.hpp>
00014
00015 namespace bsc = boost::spirit::classic;
00016
00017 namespace AIRTSP {
00018
00019     namespace ScheduleParserHelper {
00020
00021         // //////////////////////////////////////
00022         // Semantic actions
00023         // //////////////////////////////////////
00024
00025         ParserSemanticAction::
00026         ParserSemanticAction (FlightPeriodStruct& ioFlightPeriod)
00027             : _flightPeriod (ioFlightPeriod) {
00028         }
00029
00030         // //////////////////////////////////////
00031         storeAirlineCode::
00032         storeAirlineCode (FlightPeriodStruct& ioFlightPeriod)
00033             : ParserSemanticAction (ioFlightPeriod) {
00034         }
00035
00036         // //////////////////////////////////////
00037         void storeAirlineCode::operator() (iterator_t iStr,
00038             iterator_t iStrEnd) const {
00039             const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00040             _flightPeriod._airlineCode = lAirlineCode;
00041             //STDAIR_LOG_DEBUG ("Airline code: " << 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             //STDAIR_LOG_DEBUG ("Flight number: " << iNumber);
00058         }
00059
00060         // //////////////////////////////////////
00061         storeDateRangeStart::
00062         storeDateRangeStart (FlightPeriodStruct& ioFlightPeriod)
00063             : ParserSemanticAction (ioFlightPeriod) {
00064         }
00065

```

```

00066 ///////////////////////////////////////////////////////////////////
00067 void storeDateRangeStart::operator() (iterator_t iStr,
00068                                     iterator_t iStrEnd) const {
00069     _flightPeriod._dateRangeStart = _flightPeriod.getDate();
00070
00071     // Reset the number of seconds
00072     _flightPeriod._itSeconds = 0;
00073 }
00074
00075 ///////////////////////////////////////////////////////////////////
00076 storeDateRangeEnd::
00077 storeDateRangeEnd (FlightPeriodStruct& ioFlightPeriod)
00078     : ParserSemanticAction (ioFlightPeriod) {
00079 }
00080
00081 ///////////////////////////////////////////////////////////////////
00082 void storeDateRangeEnd::operator() (iterator_t iStr,
00083                                     iterator_t iStrEnd) const {
00084     // As a Boost date period (DatePeriod_T) defines the last day of
00085     // the period to be end-date - one day, we have to add one day to that
00086     // end date before.
00087     const stdair::DateOffset_T oneDay (1);
00088     _flightPeriod._dateRangeEnd = _flightPeriod.getDate() + oneDay;
00089
00090     // Transform the date pair (i.e., the date range) into a date period
00091     _flightPeriod._dateRange =
00092         stdair::DatePeriod_T (_flightPeriod._dateRangeStart,
00093                               _flightPeriod._dateRangeEnd);
00094
00095     // Reset the number of seconds
00096     _flightPeriod._itSeconds = 0;
00097
00098     // Set the (default) operating airline and flight number
00099     _flightPeriod._itLeg._airlineCode = _flightPeriod._airlineCode;
00100     _flightPeriod._itLeg._flightNumber = _flightPeriod._flightNumber;
00101 }
00102
00103 ///////////////////////////////////////////////////////////////////
00104 storeDow::storeDow (FlightPeriodStruct& ioFlightPeriod)
00105     : ParserSemanticAction (ioFlightPeriod) {
00106 }
00107
00108 ///////////////////////////////////////////////////////////////////
00109 void storeDow::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00110     stdair::DOW_String_T lDow (iStr, iStrEnd);
00111     _flightPeriod._dow = lDow;
00112     //STDAIR_LOG_DEBUG ("DOW: " << lDow);
00113 }
00114
00115 ///////////////////////////////////////////////////////////////////
00116 storeLegBoardingPoint::
00117 storeLegBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00118     : ParserSemanticAction (ioFlightPeriod) {
00119 }
00120
00121 ///////////////////////////////////////////////////////////////////
00122 void storeLegBoardingPoint::operator() (iterator_t iStr,
00123                                         iterator_t iStrEnd) const {
00124     stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00125
00126     // If a leg has already been parsed, add it to the FlightPeriod
00127     if (_flightPeriod._legAlreadyDefined == true) {
00128         _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00129     } else {
00130         _flightPeriod._legAlreadyDefined = true;
00131     }
00132 }

```

```

00133         // Set the (new) boarding point
00134         _flightPeriod._itLeg._boardingPoint = lBoardingPoint;
00135
00136         // As that's the beginning of a new leg, the list of cabins
00137         // must be reset
00138         _flightPeriod._itLeg._cabinList.clear();
00139
00140         // Add the airport code if it is not already stored in the airport lists
00141         _flightPeriod.addAirport (lBoardingPoint);
00142     }
00143
00144     // //////////////////////////////////////
00145     storeLegOffPoint::
00146     storeLegOffPoint (FlightPeriodStruct& ioFlightPeriod)
00147         : ParserSemanticAction (ioFlightPeriod) {
00148     }
00149
00150     // //////////////////////////////////////
00151     void storeLegOffPoint::operator() (iterator_t iStr,
00152                                       iterator_t iStrEnd) const {
00153         stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00154         _flightPeriod._itLeg._offPoint = lOffPoint;
00155
00156         // Add the airport code if it is not already stored in the airport lists
00157         _flightPeriod.addAirport (lOffPoint);
00158     }
00159
00160     // //////////////////////////////////////
00161     storeOperatingAirlineCode::
00162     storeOperatingAirlineCode (FlightPeriodStruct& ioFlightPeriod)
00163         : ParserSemanticAction (ioFlightPeriod) {
00164     }
00165
00166     // //////////////////////////////////////
00167     void storeOperatingAirlineCode::operator() (iterator_t iStr,
00168                                                iterator_t iStrEnd) const {
00169         const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00170         if (lAirlineCode.size() == 2) {
00171             _flightPeriod._itLeg._airlineCode = lAirlineCode;
00172         }
00173
00174         //STDAIR_LOG_DEBUG ("Airline code: " << lAirlineCode);
00175     }
00176
00177     // //////////////////////////////////////
00178     storeOperatingFlightNumber::
00179     storeOperatingFlightNumber (FlightPeriodStruct& ioFlightPeriod)
00180         : ParserSemanticAction (ioFlightPeriod) {
00181     }
00182
00183     // //////////////////////////////////////
00184     void storeOperatingFlightNumber::operator() (unsigned int iNumber) const {
00185         _flightPeriod._itLeg._flightNumber = iNumber;
00186         //STDAIR_LOG_DEBUG ("Flight number: " << iNumber);
00187     }
00188
00189     // //////////////////////////////////////
00190     storeBoardingTime::
00191     storeBoardingTime (FlightPeriodStruct& ioFlightPeriod)
00192         : ParserSemanticAction (ioFlightPeriod) {
00193     }
00194
00195     // //////////////////////////////////////
00196     void storeBoardingTime::operator() (iterator_t iStr,
00197                                       iterator_t iStrEnd) const {
00198         _flightPeriod._itLeg._boardingTime = _flightPeriod.getTime();
00199

```

```

00200         // Reset the number of seconds
00201         _flightPeriod._itSeconds = 0;
00202
00203         // Reset the date off-set
00204         _flightPeriod._dateOffset = 0;
00205     }
00206
00207     // ////////////////////////////////////////
00208     storeOffTime::
00209     storeOffTime (FlightPeriodStruct& ioFlightPeriod)
00210         : ParserSemanticAction (ioFlightPeriod) {
00211     }
00212
00213     // ////////////////////////////////////////
00214     void storeOffTime::operator() (iterator_t iStr,
00215                                     iterator_t iStrEnd) const {
00216         _flightPeriod._itLeg._offTime = _flightPeriod.getTime();
00217
00218         // Reset the number of seconds
00219         _flightPeriod._itSeconds = 0;
00220
00221         // As the boarding date off set is optional, it can be set only
00222         // afterwards, based on the staging date off-set value
00223         // (_flightPeriod._dateOffset).
00224         const stdair::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00225         _flightPeriod._itLeg._boardingDateOffset = lDateOffset;
00226     }
00227
00228     // ////////////////////////////////////////
00229     storeElapsedTime::
00230     storeElapsedTime (FlightPeriodStruct& ioFlightPeriod)
00231         : ParserSemanticAction (ioFlightPeriod) {
00232     }
00233
00234     // ////////////////////////////////////////
00235     void storeElapsedTime::operator() (iterator_t iStr,
00236                                         iterator_t iStrEnd) const {
00237         _flightPeriod._itLeg._elapsed = _flightPeriod.getTime();
00238
00239         // Reset the number of seconds
00240         _flightPeriod._itSeconds = 0;
00241
00242         // As the boarding date off set is optional, it can be set only
00243         // afterwards, based on the staging date off-set value
00244         // (_flightPeriod._dateOffset).
00245         const stdair::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00246         _flightPeriod._itLeg._offDateOffset = lDateOffset;
00247     }
00248
00249     // ////////////////////////////////////////
00250     storeLegCabinCode::
00251     storeLegCabinCode (FlightPeriodStruct& ioFlightPeriod)
00252         : ParserSemanticAction (ioFlightPeriod) {
00253     }
00254
00255     // ////////////////////////////////////////
00256     void storeLegCabinCode::operator() (char iChar) const {
00257         _flightPeriod._itLegCabin._cabinCode = iChar;
00258         //STDAIR_LOG_DEBUG ("Cabin code: " << iChar);
00259     }
00260
00261     // ////////////////////////////////////////
00262     storeCapacity::
00263     storeCapacity (FlightPeriodStruct& ioFlightPeriod)
00264         : ParserSemanticAction (ioFlightPeriod) {
00265     }
00266

```

```

00267 // //////////////////////////////////////
00268 void storeCapacity::operator() (double iReal) const {
00269     _flightPeriod._itLegCabin._capacity = iReal;
00270     //STDAIR_LOG_DEBUG ("Capacity: " << iReal);
00271
00272     // The capacity is the last (according to the arrival order
00273     // within the schedule input file) detail of the leg cabin. Hence,
00274     // when a capacity is parsed, it means that the full cabin
00275     // details have already been parsed as well: the cabin can
00276     // thus be added to the leg.
00277     _flightPeriod._itLeg._cabinList.push_back (_flightPeriod._itLegCabin);
00278 }
00279
00280 // //////////////////////////////////////
00281 storeSegmentSpecificity::
00282 storeSegmentSpecificity (FlightPeriodStruct& ioFlightPeriod)
00283     : ParserSemanticAction (ioFlightPeriod) {
00284 }
00285
00286 // //////////////////////////////////////
00287 void storeSegmentSpecificity::operator() (char iChar) const {
00288     if (iChar == '0') {
00289         _flightPeriod._areSegmentDefinitionsSpecific = false;
00290     } else {
00291         _flightPeriod._areSegmentDefinitionsSpecific = true;
00292     }
00293
00294     // Do a few sanity checks: the two lists should get exactly the same
00295     // content (in terms of airport codes). The only difference is that one
00296     // is a STL set, and the other a STL vector.
00297     assert (_flightPeriod._airportList.size()
00298             == _flightPeriod._airportOrderedList.size());
00299     assert (_flightPeriod._airportList.size() >= 2);
00300
00301     // Since all the legs have now been parsed, we get all the airports
00302     // and the segments may be built.
00303     _flightPeriod.buildSegments();
00304 }
00305
00306 // //////////////////////////////////////
00307 storeSegmentBoardingPoint::
00308 storeSegmentBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00309     : ParserSemanticAction (ioFlightPeriod) {
00310 }
00311
00312 // //////////////////////////////////////
00313 void storeSegmentBoardingPoint::operator() (iterator_t iStr,
00314                                             iterator_t iStrEnd) const {
00315     stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00316     _flightPeriod._itSegment._boardingPoint = lBoardingPoint;
00317 }
00318
00319 // //////////////////////////////////////
00320 storeSegmentOffPoint::
00321 storeSegmentOffPoint (FlightPeriodStruct& ioFlightPeriod)
00322     : ParserSemanticAction (ioFlightPeriod) {
00323 }
00324
00325 // //////////////////////////////////////
00326 void storeSegmentOffPoint::operator() (iterator_t iStr,
00327                                       iterator_t iStrEnd) const {
00328     stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00329     _flightPeriod._itSegment._offPoint = lOffPoint;
00330 }
00331
00332 // //////////////////////////////////////
00333 storeSegmentCabinCode::

```

```

00334     storeSegmentCabinCode (FlightPeriodStruct& ioFlightPeriod)
00335         : ParserSemanticAction (ioFlightPeriod) {
00336     }
00337
00338     // //////////////////////////////////////
00339 void storeSegmentCabinCode::operator() (char iChar) const {
00340     _flightPeriod._itSegmentCabin._cabinCode = iChar;
00341 }
00342
00343     // //////////////////////////////////////
00344 storeClasses::
00345 storeClasses (FlightPeriodStruct& ioFlightPeriod)
00346     : ParserSemanticAction (ioFlightPeriod) {
00347 }
00348
00349     // //////////////////////////////////////
00350 void storeClasses::operator() (iterator_t iStr,
00351                               iterator_t iStrEnd) const {
00352     std::string lClasses (iStr, iStrEnd);
00353     _flightPeriod._itSegmentCabin._classes = lClasses;
00354
00355     // The list of classes is the last (according to the arrival order
00356     // within the schedule input file) detail of the segment cabin. Hence,
00357     // when a list of classes is parsed, it means that the full segment
00358     // cabin details have already been parsed as well: the segment cabin
00359     // can thus be added to the segment.
00360     if (_flightPeriod._areSegmentDefinitionsSpecific == true) {
00361         _flightPeriod.addSegmentCabin (_flightPeriod._itSegment,
00362                                       _flightPeriod._itSegmentCabin);
00363     } else {
00364         _flightPeriod.addSegmentCabin (_flightPeriod._itSegmentCabin);
00365     }
00366 }
00367
00368     // //////////////////////////////////////
00369 storeFamilyCode::
00370 storeFamilyCode (FlightPeriodStruct& ioFlightPeriod)
00371     : ParserSemanticAction (ioFlightPeriod) {
00372 }
00373
00374     // //////////////////////////////////////
00375 void storeFamilyCode::operator() (int iCode) const {
00376     std::ostringstream ostr;
00377     ostr << iCode;
00378     _flightPeriod._itSegmentCabin._itFamilyCode = ostr.str();
00379 }
00380
00381     // //////////////////////////////////////
00382 storeFRAT5CurveKey::
00383 storeFRAT5CurveKey (FlightPeriodStruct& ioFlightPeriod)
00384     : ParserSemanticAction (ioFlightPeriod) {
00385 }
00386
00387     // //////////////////////////////////////
00388 void storeFRAT5CurveKey::operator() (iterator_t iStr,
00389                                     iterator_t iStrEnd) const {
00390     const std::string lKey (iStr, iStrEnd);
00391     _flightPeriod._itSegmentCabin._itFRAT5CurveKey = lKey;
00392     //STDAIR_LOG_DEBUG ("FRAT5 key: " << lKey);
00393 }
00394
00395     // //////////////////////////////////////
00396 storeFFDisutilityCurveKey::
00397 storeFFDisutilityCurveKey (FlightPeriodStruct& ioFlightPeriod)
00398     : ParserSemanticAction (ioFlightPeriod) {
00399 }
00400

```

```

00401 // //////////////////////////////////////
00402 void storeFFDisutilityCurveKey::operator() (iterator_t iStr,
00403                                             iterator_t iStrEnd) const {
00404     const std::string lKey (iStr, iStrEnd);
00405     _flightPeriod._itSegmentCabin._itFFDisutilityCurveKey = lKey;
00406 }
00407
00408 // //////////////////////////////////////
00409 storeFClasses::
00410 storeFClasses (FlightPeriodStruct& ioFlightPeriod)
00411 : ParserSemanticAction (ioFlightPeriod) {
00412 }
00413
00414 // //////////////////////////////////////
00415 void storeFClasses::operator() (iterator_t iStr,
00416                                 iterator_t iStrEnd) const {
00417     std::string lClasses (iStr, iStrEnd);
00418     FareFamilyStruct lFareFamily (_flightPeriod._itSegmentCabin._itFamilyCode,
00419                                   _flightPeriod._itSegmentCabin.
00420                                   _itFRAT5CurveKey,
00421                                   _flightPeriod._itSegmentCabin.
00422                                   _itFFDisutilityCurveKey,
00423                                   lClasses);
00424
00425     // The list of classes is the last (according to the arrival order
00426     // within the schedule input file) detail of the segment cabin. Hence,
00427     // when a list of classes is parsed, it means that the full segment
00428     // cabin details have already been parsed as well: the segment cabin
00429     // can thus be added to the segment.
00430     if (_flightPeriod._areSegmentDefinitionsSpecific == true) {
00431         _flightPeriod.addFareFamily (_flightPeriod._itSegment,
00432                                     _flightPeriod._itSegmentCabin,
00433                                     lFareFamily);
00434     } else {
00435         _flightPeriod.addFareFamily (_flightPeriod._itSegmentCabin,
00436                                     lFareFamily);
00437     }
00438 }
00439
00440 // //////////////////////////////////////
00441 doEndFlight::
00442 doEndFlight (stdair::BomRoot& ioBomRoot,
00443              FlightPeriodStruct& ioFlightPeriod)
00444 : ParserSemanticAction (ioFlightPeriod),
00445   _bomRoot (ioBomRoot) {
00446 }
00447
00448 // void doEndFlight::operator() (char iChar) const {
00449 void doEndFlight::operator() (iterator_t iStr,
00450                               iterator_t iStrEnd) const {
00451     assert (_flightPeriod._legAlreadyDefined == true);
00452     _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00453
00454     // The lists of legs and cabins must be reset
00455     _flightPeriod._legAlreadyDefined = false;
00456     _flightPeriod._itLeg._cabinList.clear();
00457
00458     // DEBUG: Display the result
00459     STDAIR_LOG_DEBUG ("FlightPeriod: " << _flightPeriod.describe());
00460
00461     // Create the FlightPeriod BOM objects, and potentially the intermediary
00462     // objects (e.g., Inventory).
00463     InventoryGenerator::createFlightPeriod (_bomRoot, _flightPeriod);
00464 }
00465

```

```

00466
00467 // //////////////////////////////////////
00468 //
00469 //   Utility Parsers
00470 //
00471 // //////////////////////////////////////
00473 int1_p_t int1_p;
00474
00476 uint2_p_t uint2_p;
00477
00479 uint4_p_t uint4_p;
00480
00482 uint1_4_p_t uint1_4_p;
00483
00485 repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3);
00486
00488 bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u);
00489
00491 bounded4_p_t year_p (uint4_p.derived(), 2000u, 2099u);
00492
00494 bounded2_p_t month_p (uint2_p.derived(), 1u, 12u);
00495
00497 bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00498
00500 repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7);
00501
00503 repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3);
00504
00506 bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u);
00507
00509 bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u);
00510
00512 bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u);
00513
00515 chset_t cabin_code_p ("A-Z");
00516
00518 int1_p_t family_code_p;
00519
00521 repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10);
00522
00524 repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26);
00525
00526
00527 // //////////////////////////////////////
00528 //   (Boost Spirit) Grammar Definition
00529 //   //////////////////////////////////////
00530
00531 // //////////////////////////////////////
00532 FlightPeriodParser::
00533 FlightPeriodParser (stdair::BomRoot& ioBomRoot,
00534                     FlightPeriodStruct& ioFlightPeriod)
00535 : _bomRoot (ioBomRoot),
00536   _flightPeriod (ioFlightPeriod) {
00537 }
00538
00539 // //////////////////////////////////////
00540 template<typename ScannerT>
00541 FlightPeriodParser::definition<ScannerT>::
00542 definition (FlightPeriodParser const& self) {
00543
00544     flight_period_list = *(not_to_be_parsed
00545                          | flight_period )
00546
00547     ;
00548
00549     not_to_be_parsed = bsc::
00549         lexeme_d[bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00550                 | bsc::eol_p];

```



```

00551
00552     flight_period = flight_key
00553     >> + ( ';' >> leg )
00554     >> ';' >> segment_section
00555     >> flight_period_end[doEndFlight (self._bomRoot, self._flightPeriod)]
00556     ;
00557
00558     flight_period_end =
00559     bsc::ch_p(';' )
00560     ;
00561
00562     flight_key = airline_code
00563     >> ';' >> flight_number
00564     >> ';' >> date[storeDateRangeStart(self._flightPeriod)]
00565     >> ';' >> date[storeDateRangeEnd(self._flightPeriod)]
00566     >> ';' >> dow[storeDow(self._flightPeriod)]
00567     ;
00568
00569     airline_code =bsc::
00570     lexeme_d[ (airline_code_p) [storeAirlineCode(self._flightPeriod)] ]
00571     ;
00572
00573     flight_number =bsc::
00574     lexeme_d[ (flight_number_p) [storeFlightNumber(self._flightPeriod)] ]
00575     ;
00576
00577     date =bsc::
00578     lexeme_d[ (year_p) [bsc::assign_a(self._flightPeriod._itYear)]
00579     >> '-'
00580     >> (month_p) [bsc::assign_a(self._flightPeriod._itMonth)]
00581     >> '-'
00582     >> (day_p) [bsc::assign_a(self._flightPeriod._itDay)]
00583     ]
00584     ;
00585
00586     dow =bsc::lexeme_d[ dow_p ]
00587     ;
00588
00589     leg = ! ( operating_leg_details >> ';' )
00590     >> leg_key
00591     >> ';' >> leg_details
00592     >> + ( ';' >> leg_cabin_details )
00593     ;
00594
00595     leg_key = (airport_p) [storeLegBoardingPoint(self._flightPeriod)]
00596     >> ';'
00597     >> (airport_p) [storeLegOffPoint(self._flightPeriod)]
00598     ;
00599
00600     operating_leg_details =
00601     bsc::lexeme_d[ (airline_code_p) [storeOperatingAirlineCode(self.
00602     _flightPeriod)] ]
00603     >> ";"
00604     >> bsc::lexeme_d[ (flight_number_p) [storeOperatingFlightNumber(self.
00605     _flightPeriod)] ]
00606     ;
00607
00608     leg_details =
00609     time[storeBoardingTime(self._flightPeriod)]
00610     >> !(date_offset)
00611     >> ';'
00612     >> time[storeOffTime(self._flightPeriod)]
00613     >> !(date_offset)
00614     >> ';'
00615     >> time[storeElapsedTime(self._flightPeriod)]
00616     ;

```

```

00616         time =bsc::
00617             lexeme_d[(hours_p)[bsc::assign_a(self._flightPeriod._itHours)]]
00618             >> ':'
00619             >> (minutes_p)[bsc::assign_a(self._flightPeriod._itMinutes)]
00620             >> !(':'
00621                 >> (seconds_p)[bsc::assign_a(self._flightPeriod._itSeconds)
00622             ])
00623         ;
00624
00625         date_offset =bsc::ch_p('/')
00626         >> (intl_p)[bsc::assign_a(self._flightPeriod._dateOffset)]
00627         ;
00628
00629         leg_cabin_details = (cabin_code_p)[storeLegCabinCode(self._flightPeriod)]
00630         >> ';' >> (bsc::ureal_p)[storeCapacity(self._flightPeriod)]
00631         ;
00632
00633         segment_key =
00634             (airport_p)[storeSegmentBoardingPoint(self._flightPeriod)]
00635             >> ';'
00636             >> (airport_p)[storeSegmentOffPoint(self._flightPeriod)]
00637             ;
00638
00639         segment_section =
00640             generic_segment | specific_segment_list
00641             ;
00642
00643         generic_segment =bsc::
00644             ch_p('0')[storeSegmentSpecificity(self._flightPeriod)]
00645             >> +(';') >> segment_cabin_details
00646             ;
00647
00648         specific_segment_list =bsc::
00649             ch_p('1')[storeSegmentSpecificity(self._flightPeriod)]
00650             >> +(';') >> segment_key >> full_segment_cabin_details
00651             ;
00652
00653         full_segment_cabin_details =
00654             +(';') >> segment_cabin_details
00655             ;
00656
00657         segment_cabin_details =
00658             (cabin_code_p)[storeSegmentCabinCode(self._flightPeriod)]
00659             >> ';' >> (class_code_list_p)[storeClasses(self._flightPeriod)]
00660             >> *(';') >> family_cabin_details
00661             ;
00662
00663         family_cabin_details =
00664             (family_code_p)[storeFamilyCode(self._flightPeriod)]
00665             >> ';'
00666             >> (key_p)[storeFRAT5CurveKey(self._flightPeriod)]
00667             >> ';'
00668             >> (key_p)[storeFFDisutilityCurveKey(self._flightPeriod)]
00669             >> ';'
00670             >> (class_code_list_p)[storeFCClasses(self._flightPeriod)]
00671             ;
00672
00673         // BOOST_SPIRIT_DEBUG_NODE (FlightPeriodParser);
00674         BOOST_SPIRIT_DEBUG_NODE (flight_period_list);
00675         BOOST_SPIRIT_DEBUG_NODE (flight_period);
00676         BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
00677         BOOST_SPIRIT_DEBUG_NODE (flight_period_end);
00678         BOOST_SPIRIT_DEBUG_NODE (flight_key);
00679         BOOST_SPIRIT_DEBUG_NODE (airline_code);
00680         BOOST_SPIRIT_DEBUG_NODE (flight_number);
00681         BOOST_SPIRIT_DEBUG_NODE (date);

```

```

00682     BOOST_SPIRIT_DEBUG_NODE (dow);
00683     BOOST_SPIRIT_DEBUG_NODE (leg);
00684     BOOST_SPIRIT_DEBUG_NODE (leg_key);
00685     BOOST_SPIRIT_DEBUG_NODE (leg_details);
00686     BOOST_SPIRIT_DEBUG_NODE (time);
00687     BOOST_SPIRIT_DEBUG_NODE (date_offset);
00688     BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
00689     BOOST_SPIRIT_DEBUG_NODE (segment_section);
00690     BOOST_SPIRIT_DEBUG_NODE (segment_key);
00691     BOOST_SPIRIT_DEBUG_NODE (generic_segment);
00692     BOOST_SPIRIT_DEBUG_NODE (specific_segment_list);
00693     BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
00694     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
00695     BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
00696 }
00697
00698 // //////////////////////////////////////
00699 template<typename ScannerT>
00700 bsc::rule<ScannerT> const&
00701 FlightPeriodParser::definition<ScannerT>::start() const {
00702     return flight_period_list;
00703 }
00704
00705 }
00706
00707 //
00708 // Entry class for the file parser
00709 //
00710 // //////////////////////////////////////
00711 FlightPeriodFileParser::
00712 FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00713                         const stdair::Filename_T& iFilename)
00714 : _filename (iFilename), _bomRoot (ioBomRoot) {
00715     init();
00716 }
00717
00718 // //////////////////////////////////////
00719 void FlightPeriodFileParser::init() {
00720     // Check that the file exists and is readable
00721     const bool doesExistAndIsReadable =
00722         stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00723
00724     if (doesExistAndIsReadable == false) {
00725         STDAIR_LOG_ERROR ("The schedule file " << _filename
00726             << " does not exist or can not be read.");
00727
00728         throw ScheduleInputFileNotFoundException ("The schedule file " + _filename
00729             + " does not exist or can not be
00730 read");
00731     }
00732
00733     // Open the file
00734     _startIterator = iterator_t (_filename);
00735
00736     // Check the filename exists and can be open
00737     if (!_startIterator) {
00738         STDAIR_LOG_ERROR ("The schedule file " << _filename << " can not be open."
00739             << std::endl);
00740
00741         throw ScheduleInputFileNotFoundException ("The file " + _filename
00742             + " does not exist or can not be
00743 read");
00744     }
00745
00746     // Create an EOF iterator

```

```
00749     _endIterator = _startIterator.make_end();
00750 }
00751
00752 // //////////////////////////////////////
00753 bool FlightPeriodFileParser::generateInventories () {
00754     bool oResult = false;
00755
00756     STDAIR_LOG_DEBUG ("Parsing schedule input file: " << _filename);
00757
00758     // Initialise the parser (grammar) with the helper/staging structure.
00759     ScheduleParserHelper::FlightPeriodParser lFPParser (_bomRoot,
00760                                                         _flightPeriod);
00761
00762     // Launch the parsing of the file and, thanks to the doEndFlight
00763     // call-back structure, the building of the whole BomRoot BOM
00764     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00765     bsc::parse_info<iterator_t> info =
00766         bsc::parse (_startIterator, _endIterator, lFPParser,
00767                     bsc::space_p - bsc::eol_p);
00768
00769     // Retrieves whether or not the parsing was successful
00770     oResult = info.hit;
00771
00772     const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
00773     if (oResult == true) {
00774         STDAIR_LOG_DEBUG ("Parsing of schedule input file: " << _filename
00775                             << " succeeded: read " << info.length
00776                             << " characters. The input file has "
00777                             << hasBeenFullyReadStr
00778                             << "been fully read. Stop point: " << info.stop);
00779     } else {
00780         // TODO: decide whether to throw an exception
00781         STDAIR_LOG_ERROR ("Parsing of schedule input file: " << _filename
00782                             << " failed: read " << info.length
00783                             << " characters. The input file has "
00784                             << hasBeenFullyReadStr
00785                             << "been fully read. Stop point: " << info.stop);
00786     }
00787 }
00788
00789 return oResult;
00790 }
00791
00792 }
```

## 26.109 airtsp/command/ScheduleParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
#include <airtsp/basic/BasParserTypes.hpp>
#include <airtsp/bom/FlightPeriodStruct.hpp>
```

### Classes

- struct [AIRTSP::ScheduleParserHelper::ParserSemanticAction](#)
- struct [AIRTSP::ScheduleParserHelper::storeAirlineCode](#)
- struct [AIRTSP::ScheduleParserHelper::storeFlightNumber](#)
- struct [AIRTSP::ScheduleParserHelper::storeDateRangeStart](#)
- struct [AIRTSP::ScheduleParserHelper::storeDateRangeEnd](#)
- struct [AIRTSP::ScheduleParserHelper::storeDow](#)
- struct [AIRTSP::ScheduleParserHelper::storeLegBoardingPoint](#)
- struct [AIRTSP::ScheduleParserHelper::storeLegOffPoint](#)
- struct [AIRTSP::ScheduleParserHelper::storeOperatingAirlineCode](#)
- struct [AIRTSP::ScheduleParserHelper::storeOperatingFlightNumber](#)
- struct [AIRTSP::ScheduleParserHelper::storeBoardingTime](#)
- struct [AIRTSP::ScheduleParserHelper::storeOffTime](#)
- struct [AIRTSP::ScheduleParserHelper::storeElapsedTime](#)
- struct [AIRTSP::ScheduleParserHelper::storeLegCabinCode](#)
- struct [AIRTSP::ScheduleParserHelper::storeCapacity](#)
- struct [AIRTSP::ScheduleParserHelper::storeSegmentSpecificity](#)
- struct [AIRTSP::ScheduleParserHelper::storeSegmentBoardingPoint](#)
- struct [AIRTSP::ScheduleParserHelper::storeSegmentOffPoint](#)
- struct [AIRTSP::ScheduleParserHelper::storeSegmentCabinCode](#)
- struct [AIRTSP::ScheduleParserHelper::storeClasses](#)
- struct [AIRTSP::ScheduleParserHelper::storeFamilyCode](#)
- struct [AIRTSP::ScheduleParserHelper::storeFRAT5CurveKey](#)
- struct [AIRTSP::ScheduleParserHelper::storeFFDisutilityCurveKey](#)
- struct [AIRTSP::ScheduleParserHelper::storeFCclasses](#)
- struct [AIRTSP::ScheduleParserHelper::doEndFlight](#)
- struct [AIRTSP::ScheduleParserHelper::FlightPeriodParser](#)
- struct [AIRTSP::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >](#)
- class [AIRTSP::FlightPeriodFileParser](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)
- namespace [AIRTSP::ScheduleParserHelper](#)

**26.110 ScheduleParserHelper.hpp**

```

00001 #ifndef __AIRTSP_CMD_SCHEDULEPARSERHELPER_HPP
00002 #define __AIRTSP_CMD_SCHEDULEPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // AirTSP
00012 #include <airtsp/AIRTSP_Types.hpp>
00013 #include <airtsp/basic/BasParserTypes.hpp>
00014 #include <airtsp/bom/FlightPeriodStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class BomRoot;
00019 }
00020
00021 namespace AIRTSP {
00022
00023     namespace ScheduleParserHelper {
00024
00025         // //////////////////////////////////////
00026         // Semantic actions
00027         // //////////////////////////////////////
00029         struct ParserSemanticAction {
00031             ParserSemanticAction (FlightPeriodStruct&);
00033             FlightPeriodStruct& _flightPeriod;
00034         };
00035
00037         struct storeAirlineCode : public ParserSemanticAction {
00039             storeAirlineCode (FlightPeriodStruct&);
00041             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00042         };
00043
00045         struct storeFlightNumber : public ParserSemanticAction {
00047             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 storeOperatingAirlineCode : public ParserSemanticAction {
00095         storeOperatingAirlineCode (FlightPeriodStruct&);
00097         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00098     };
00099
00101     struct storeOperatingFlightNumber : public ParserSemanticAction {
00103         storeOperatingFlightNumber (FlightPeriodStruct&);
00105         void operator() (unsigned int iNumber) const;
00106     };
00107
00109     struct storeBoardingTime : public ParserSemanticAction {
00111         storeBoardingTime (FlightPeriodStruct&);
00113         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00114     };
00115
00117     struct storeOffTime : public ParserSemanticAction {
00119         storeOffTime (FlightPeriodStruct&);
00121         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00122     };
00123
00125     struct storeElapsedTime : public ParserSemanticAction {
00127         storeElapsedTime (FlightPeriodStruct&);
00129         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00130     };
00131
00133     struct storeLegCabinCode : public ParserSemanticAction {
00135         storeLegCabinCode (FlightPeriodStruct&);
00137         void operator() (char iChar) const;
00138     };
00139
00141     struct storeCapacity : public ParserSemanticAction {
00143         storeCapacity (FlightPeriodStruct&);
00145         void operator() (double iReal) const;
00146     };
00147
00152     struct storeSegmentSpecificity : public ParserSemanticAction {
00154         storeSegmentSpecificity (FlightPeriodStruct&);
00156         void operator() (char iChar) const;
00157     };
00158
00160     struct storeSegmentBoardingPoint : public ParserSemanticAction {
00162         storeSegmentBoardingPoint (FlightPeriodStruct&);
00164         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00165     };
00166
00168     struct storeSegmentOffPoint : public ParserSemanticAction {
00170         storeSegmentOffPoint (FlightPeriodStruct&);
00172         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00173     };
00174
00176     struct storeSegmentCabinCode : public ParserSemanticAction {
00178         storeSegmentCabinCode (FlightPeriodStruct&);
00180         void operator() (char iChar) const;
00181     };
00182
00184     struct storeClasses : public ParserSemanticAction {
00186         storeClasses (FlightPeriodStruct&);
00188         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00189     };
00190
00192     struct storeFamilyCode : public ParserSemanticAction {
00194         storeFamilyCode (FlightPeriodStruct&);
00196         void operator() (int iCode) const;
00197     };
00198
```

```

00200     struct storeFRAT5CurveKey : public ParserSemanticAction {
00202         storeFRAT5CurveKey (FlightPeriodStruct&);
00204         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00205     };
00206
00208     struct storeFFDisutilityCurveKey : public ParserSemanticAction {
00210         storeFFDisutilityCurveKey (FlightPeriodStruct&);
00212         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00213     };
00214
00216     struct storeFClasses : public ParserSemanticAction {
00218         storeFClasses (FlightPeriodStruct&);
00220         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00221     };
00222
00224     struct doEndFlight : public ParserSemanticAction {
00226         doEndFlight (stdair::BomRoot&, FlightPeriodStruct&);
00228         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00230         stdair::BomRoot& _bomRoot;
00231     };
00232
00233
00235     //
00236     // (Boost Spirit) Grammar Definition
00237     //
00239
00281     struct FlightPeriodParser :
00282         public boost::spirit::classic::grammar<FlightPeriodParser> {
00283
00284         FlightPeriodParser (stdair::BomRoot&, FlightPeriodStruct&);
00285
00286         template <typename ScannerT>
00287         struct definition {
00288             definition (FlightPeriodParser const& self);
00289
00290             // Instantiation of rules
00291             boost::spirit::classic::rule<ScannerT> flight_period_list, flight_period,
00292
00293             not_to_be_parsed, flight_period_end, flight_key, airline_code,
00294             flight_number, date, dow, time, date_offset,
00295             leg, leg_key, operating_leg_details, leg_details, leg_cabin_details,
00296             segment_section, segment_key, full_segment_cabin_details,
00297             segment_cabin_details, full_family_cabin_details,
00298             family_cabin_details, generic_segment, specific_segment_list;
00299
00300             boost::spirit::classic::rule<ScannerT> const& start() const;
00301         };
00302
00303         // Parser Context
00304         stdair::BomRoot& _bomRoot;
00305         FlightPeriodStruct& _flightPeriod;
00306     };
00307
00308 }
00313
00314 //
00315 // Entry class for the file parser
00316 //
00318
00323     class FlightPeriodFileParser : public stdair::CmdAbstract {
00324     public:
00326         FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00327                                 const stdair::Filename_T& iFilename);
00328
00330         bool generateInventories ();
00331     private:
00332

```



```
00334     void init();
00335
00336 private:
00337     // Attributes
00339     stdair::Filename_T _filename;
00340
00342     iterator_t _startIterator;
00343
00345     iterator_t _endIterator;
00346
00348     stdair::BomRoot& _bomRoot;
00349
00351     FlightPeriodStruct _flightPeriod;
00352 };
00353
00354 }
00355 #endif // __AIRTSP_CMD_SCHEDULEPARSERHELPER_HPP
```

## 26.111 airtsp/command/SegmentPathGenerator.cpp File Reference

```
#include <cassert>
#include <vector>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightPeriod.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/bom/ReachableUniverse.hpp>
#include <airtsp/bom/OriginDestinationSet.hpp>
#include <airtsp/bom/SegmentPathPeriod.hpp>
#include <airtsp/command/SegmentPathGenerator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.112 SegmentPathGenerator.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <vector>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/bom/Inventory.hpp>
00012 #include <stdair/bom/FlightPeriod.hpp>
00013 #include <stdair/bom/SegmentPeriod.hpp>
00014 #include <stdair/factory/FacBomManager.hpp>
00015 #include <stdair/service/Logger.hpp>
00016 // AirTSP
00017 #include <airtsp/bom/ReachableUniverse.hpp>
00018 #include <airtsp/bom/OriginDestinationSet.hpp>
00019 #include <airtsp/bom/SegmentPathPeriod.hpp>
00020 #include <airtsp/command/SegmentPathGenerator.hpp>
00021
00022 namespace AIRTSP {
00023
00024 // //////////////////////////////////////
00025 void SegmentPathGenerator::
00026 createSegmentPathNetwork (const stdair::BomRoot& iBomRoot) {
00027
00028     // Build the list of single-segment segment path objects.
00029     const stdair::InventoryList_T& lInventoryList =
00030         stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00031     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
00032          itInv != lInventoryList.end(); ++itInv) {
00033         const stdair::Inventory* lCurrentInventory_ptr = *itInv;
00034         assert (lCurrentInventory_ptr != NULL);
00035
00036         //
00037         createSinglePaths (*lCurrentInventory_ptr);
00038     }
00039
00040     // Build the list of i-fixed-length segment path objects. In other words,
00041     // build the whole segment path network.
00042     for (stdair::NbOfSegments_T i = 2;
00043          i <= stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND; ++i) {
00044         buildSegmentPathNetwork (iBomRoot, i);
00045     }
00046 }
00047
00048 // //////////////////////////////////////
00049 void SegmentPathGenerator::
00050 createSinglePaths (const stdair::Inventory& iInventory) {
00051
00052     const stdair::FlightPeriodList_T& lFlightPeriodList =
00053         stdair::BomManager::getList<stdair::FlightPeriod> (iInventory);
00054     for (stdair::FlightPeriodList_T::const_iterator itFlightPeriod =
00055          lFlightPeriodList.begin();
00056          itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
00057         const stdair::FlightPeriod* lCurrentFlightPeriod_ptr = *itFlightPeriod;
00058         assert (lCurrentFlightPeriod_ptr != NULL);
00059
00060         //
00061         createSinglePaths (*lCurrentFlightPeriod_ptr);
00062     }
00063 }
00064
00065 // //////////////////////////////////////

```

```

00066 void SegmentPathGenerator::
00067 createSinglePaths (const stdair::FlightPeriod& iFlightPeriod) {
00068
00069     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00070         stdair::BomManager::getList<stdair::SegmentPeriod> (iFlightPeriod);
00071     for (stdair::SegmentPeriodList_T::const_iterator itSegmentPeriod =
00072         lSegmentPeriodList.begin();
00073         itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
00074         stdair::SegmentPeriod* lCurrentSegmentPeriod_ptr = *itSegmentPeriod;
00075         assert (lCurrentSegmentPeriod_ptr != NULL);
00076
00077         //
00078         createSinglePath (*lCurrentSegmentPeriod_ptr);
00079     }
00080 }
00081
00082 // //////////////////////////////////////
00083 void SegmentPathGenerator::
00084 createSinglePath (stdair::SegmentPeriod& ioSegmentPeriod) {
00085
00086     // Retrieve the BOM tree root
00087     const stdair::AirportCode_T& lOrigin = ioSegmentPeriod.getBoardingPoint();
00088     const stdair::FlightPeriod& lFlightPeriod =
00089         stdair::BomManager::getParent<stdair::FlightPeriod> (ioSegmentPeriod);
00090     const stdair::Inventory& lInventory =
00091         stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00092     stdair::BomRoot& lBomRoot =
00093         stdair::BomManager::getParent<stdair::BomRoot> (lInventory);
00094
00095     // Retrieve the ReachableUniverse (if existing) which corresponds
00096     // to the origin. If it does not exist, then create one.
00097     ReachableUniverse* lReachableUniverse_ptr =
00098         stdair::BomManager::getObjectPtr<ReachableUniverse> (lBomRoot, lOrigin);
00099     if (lReachableUniverse_ptr == NULL) {
00100         ReachableUniverseKey lKey (lOrigin);
00101         lReachableUniverse_ptr =
00102             &stdair::FacBom<ReachableUniverse>::instance().create (lKey);
00103         stdair::FacBomManager::addToListAndMap (lBomRoot, *lReachableUniverse_ptr);
00104
00105         stdair::FacBomManager::linkWithParent (lBomRoot, *lReachableUniverse_ptr);
00106     }
00107     assert (lReachableUniverse_ptr != NULL);
00108
00109     //
00110     createSinglePath (*lReachableUniverse_ptr, ioSegmentPeriod);
00111 }
00112 // //////////////////////////////////////
00113 void SegmentPathGenerator::
00114 createSinglePath (ReachableUniverse& ioReachableUniverse,
00115                 stdair::SegmentPeriod& ioSegmentPeriod) {
00116
00117     const stdair::AirportCode_T& lDestination = ioSegmentPeriod.getOffPoint();
00118
00119     // Retrieve the origin-destination set (if existing) which corresponds
00120     // to the destination. If it does not exist, then create one.
00121     OriginDestinationSet* lOriginDestinationSet_ptr =
00122         stdair::BomManager::getObjectPtr<OriginDestinationSet> (ioReachableUniverse,
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640

```

```

00131         stdair::FacBomManager::linkWithParent (ioReachableUniverse,
00132                                                 *lOriginDestinationSet_ptr);
00133     }
00134     assert (lOriginDestinationSet_ptr != NULL);
00135
00136     // Create a segment path period and add it to the corresponding
00137     // origin-destination set and reachable-universe.
00138     const stdair::FlightPeriod& lFlightPeriod =
00139         stdair::BomManager::getParent<stdair::FlightPeriod> (ioSegmentPeriod);
00140     const stdair::PeriodStruct& lPeriodOfFlight = lFlightPeriod.getPeriod();
00141
00142     // The departure period of the segment is the departure period of
00143     // the flight plus the boarding date offset of the segment.
00144     const stdair::DateOffset_T& lBoardingDateOffset =
00145         ioSegmentPeriod.getBoardingDateOffset();
00146
00147     const stdair::PeriodStruct lPeriodOfSegment =
00148         lPeriodOfFlight.addDateOffset (lBoardingDateOffset);
00149
00150     const stdair::Duration_T& lBoardingTime = ioSegmentPeriod.getBoardingTime();
00151     const stdair::Duration_T& lElapsed = ioSegmentPeriod.getElapsedTime();
00152
00153     DateOffsetList_T lDateOffsetList;
00154     const stdair::DateOffset_T lFirstDateOffset (0);
00155     lDateOffsetList.push_back (lFirstDateOffset);
00156
00157     const SegmentPathPeriodKey lSegmentPathKey (lPeriodOfSegment,
00158                                                 lBoardingTime, lElapsed,
00159                                                 lDateOffsetList, 1);
00160
00161     SegmentPathPeriod& lSegmentPathPeriod =
00162         stdair::FacBom<SegmentPathPeriod>::instance().create (lSegmentPathKey);
00163
00164     addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod);
00165
00166     // Link the SegmentPathPeriod object with its parent, namely
00167     // OriginDestinationSet
00168     stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
00169                                     lSegmentPathPeriod);
00170     stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
00171                                     lSegmentPathPeriod);
00172
00173     // Link the SegmentPathPeriod and SegmentPeriod objects. Note that
00174     // the SegmentPeriod object has already a parent, namely FlightPeriod.
00175     stdair::FacBomManager::addToList (lSegmentPathPeriod,
00176                                     ioSegmentPeriod);
00177 }
00178
00179 // //////////////////////////////////////
00180 void SegmentPathGenerator::
00181 addSegmentPathPeriod (ReachableUniverse& ioReachableUniverse,
00182                     const SegmentPathPeriod& iSegmentPathPeriod) {
00183
00184     const stdair::NbOfSegments_T& lNbOfSegments =
00185         iSegmentPathPeriod.getNbOfSegments();
00186
00187     assert (lNbOfSegments > 0
00188           && lNbOfSegments <= stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND);
00189
00190     // If needed, initialise the list of lists with empty fixed-length
00191     // segment path period lists.
00192
00193     SegmentPathPeriodListList_T& lSegmentPathPeriodListList =
00194         ioReachableUniverse._segmentPathPeriodListList;
00195     while (lSegmentPathPeriodListList.size() < lNbOfSegments) {
00196         SegmentPathPeriodList_T lSegmentPathPeriodList;
00197         lSegmentPathPeriodListList.push_back (lSegmentPathPeriodList);
00198     }

```

```

00204     }
00205
00206     // Retrieve the i-fixed-length segment path period list (i = number of
00207     // segments).
00208     SegmentPathPeriodLightList_T& lSegmentPathPeriodList =
00209         lSegmentPathPeriodListList.at (lNbOfSegments-1);
00210
00211     // Add the SegmentPathPeriod to that fixed-length-path list.
00212     lSegmentPathPeriodList.push_back (&iSegmentPathPeriod);
00213 }
00214
00215 // //////////////////////////////////////
00216 void SegmentPathGenerator::
00217 buildSegmentPathNetwork (const stdair::BomRoot& iBomRoot,
00218                         const stdair::NbOfSegments_T& lNbOfSegments) {
00219
00220     const ReachableUniverseList_T& lReachableUniverseList =
00221         stdair::BomManager::getList<ReachableUniverse> (iBomRoot);
00222     for (ReachableUniverseList_T::const_iterator itReachableUniverse =
00223         lReachableUniverseList.begin();
00224         itReachableUniverse != lReachableUniverseList.end();
00225         ++itReachableUniverse) {
00226         ReachableUniverse* lReachableUniverse_ptr = *itReachableUniverse;
00227         assert (lReachableUniverse_ptr != NULL);
00228
00229         //
00230         buildSegmentPathNetwork (*lReachableUniverse_ptr, lNbOfSegments);
00231     }
00232 }
00233
00234 // //////////////////////////////////////
00235 void SegmentPathGenerator::
00236 buildSegmentPathNetwork (ReachableUniverse& ioReachableUniverse,
00237                         const stdair::NbOfSegments_T& iNbOfSegments) {
00238
00239     // The goal of that method is to build the i-fixed-length
00240     // segment path period objects, knowing that all the
00241     // lower-fixed-length segment path period objects have already been
00242     // built during the previous steps. Once an i-fixed-length
00243     // segment path period object is created, it is added to the list of
00244     // the (fixed-length segment path period object) lists.
00245
00246     // Hence, at that iteration, by construction, the list of the
00247     // (fixed-length segment path period object) lists should already get
00248     // a size of i-1, if there were such possibilities (in terms of
00249     // segment path period). In that case, at the end of the method, its
00250     // size should be of i.
00251
00252     // If the size of the list of the (fixed-length segment path period
00253     // object) lists is (strictly) less than i-1, it means that that
00254     // reachable universe has no more possibilities of destinations. We
00255     // are thus done at that stage.
00256     const SegmentPathPeriodListList_T& lSegmentPathPeriodListList =
00257         ioReachableUniverse.getSegmentPathPeriodListList();
00258     const unsigned short lNbOfSegments_m1 = iNbOfSegments - 1;
00259     assert (lNbOfSegments_m1 >= 0);
00260     if (lSegmentPathPeriodListList.size() < lNbOfSegments_m1) {
00261         return;
00262     }
00263
00264     // Retrieve the (i-1)-fixed-length segment path period list (i = number of
00265     // segments).
00266
00267     // Note that a STL vector starts at 0, whereas the number of segments
00268     // starts at 1. Hence, (i-1) for the length (in number of segments)
00269     // corresponds to [iNbOfSegments-2] for the STL vector.
00270
00271

```

```

00276 // As the lSegmentPathPeriodListList may change during the next loop
00277 // iterations (as some SegmentPathPeriod objects are created and linked to
00278 // ReachableUniverse), we need to take the initial copy of that list.
00279 const SegmentPathPeriodLightList_T lSegmentPathPeriodLightList_im1 =
00280     lSegmentPathPeriodListList.at (iNbOfSegments-2);
00281
00282 // Iterate on the (i-1)-fixed-length segment path period objects, in order
00283 // to build a i-fixed-length segment path period objects.
00284 // There are two steps:
00285 // 1. Retrieve the airport-dates at a (i-1) length (in number of segments)
00286 //    of the origin airport-date.
00287 // 2. From each of such (i-1) airport-date, add the single-segment pathes
00288 //    to the (i-1)-length pathes, so as to make i-length pathes.
00289 for (SegmentPathPeriodLightList_T::const_iterator itSegmentPathPeriodList =
00290     lSegmentPathPeriodLightList_im1.begin();
00291     itSegmentPathPeriodList != lSegmentPathPeriodLightList_im1.end();
00292     ++itSegmentPathPeriodList) {
00293     const SegmentPathPeriod* lSegmentPathPeriod_im1_ptr =
00294         *itSegmentPathPeriodList;
00295     assert (lSegmentPathPeriod_im1_ptr != NULL);
00296
00297     // Get the reachable-universe departing from the destination of
00298     // the current segment path period.
00299     const stdair::AirportCode_T& lDestination_im1 =
00300         lSegmentPathPeriod_im1_ptr->getDestination();
00301     const stdair::BomRoot& lBomRoot =
00302         stdair::BomManager::getParent<stdair::BomRoot> (ioReachableUniverse);
00303     const ReachableUniverse* lReachableUniverseFromDestination_im1_ptr =
00304         stdair::BomManager::getObjectPtr<ReachableUniverse> (lBomRoot,
00305             lDestination_im1);
00306
00307     // If there is no ReachableUniverse corresponding to the destination (off
00308     // point of the last SegmentDate), it means that the destination is
00309     // an end point (no other SegmentDate is starting from there).
00310     // Hence, there is nothing else to do for now for that (final)
00311     // destination, and we can process the next (i-1)-segment path period.
00312     if (lReachableUniverseFromDestination_im1_ptr == NULL) {
00313         continue;
00314     }
00315     assert (lReachableUniverseFromDestination_im1_ptr != NULL);
00316
00317     // Retrieve the single-segment segment path period list,
00318     // so as to make a i-length SegmentPathPeriod.
00319     const SegmentPathPeriodListList_T&
00320         lSegmentPathPeriodListListFromDestination_im1 =
00321         lReachableUniverseFromDestination_im1_ptr->
00322         getSegmentPathPeriodListList();
00323     assert (lSegmentPathPeriodListListFromDestination_im1.size() >= 1);
00324
00325     // As the lSegmentPathPeriodListListFromDestination_im1 may change during
00326     // the next loop iterations (as some SegmentPathPeriod objects are
00327     // created and linked to ReachableUniverse), we need to take the initial
00328     // copy of that list.
00329     const SegmentPathPeriodLightList_T lSingleSegmentPathPeriodLightListFromDes
00330         tination_im1 =
00331         lSegmentPathPeriodListListFromDestination_im1.at (0);
00332
00333     for (SegmentPathPeriodLightList_T::const_iterator
00334         itSegmentPathPeriodFromDestination_im1 =
00335         lSingleSegmentPathPeriodLightListFromDestination_im1.begin();
00336         itSegmentPathPeriodFromDestination_im1
00337         != lSingleSegmentPathPeriodLightListFromDestination_im1.end();
00338         ++itSegmentPathPeriodFromDestination_im1) {
00339         const SegmentPathPeriod* lSingleSegmentPathPeriodFromDestination_im1_ptr=
00340
00341         *itSegmentPathPeriodFromDestination_im1;
00342         assert (lSingleSegmentPathPeriodFromDestination_im1_ptr != NULL);

```

```

00341
00342     // Check if the (i-1)-length segment path period can be fused with the
00343     // single segment segment path period in order to create an i-length
00344     // segment path period. The function will return a valid or non-valid
00345     // segment path period key.
00346
00347     // The two segment path period above can be fused (and will produce a
00348     // valid new segment path period key) if:
00349     // 1. A passenger can connect from the last segment of the
00350     // first segment path and the first segment of the next segment path.
00351     // These two segments should not create another segment.
00352     // 2. There is no circle within the new segment path.
00353     // 3. The intersection of the two periods is non-empty.
00354     SegmentPathPeriodKey lSegmentPathPeriodKey_i =
00355     lSegmentPathPeriod_iml_ptr->connectWithAnotherSegment (*lSingleSegmentP
athPeriodFromDestination_iml_ptr);
00356
00357     if (lSegmentPathPeriodKey_i.isValid () == false) {
00358         continue;
00359     }
00360
00361     // Get the off point of the single-segment SegmentPathPeriod
00362     // attached to the intermediate destination (iml). That off point is
00363     // at a length i of the initial ReachableUniverse: (i-1) + 1.
00364     const stdair::AirportCode_T& lDestination_i =
00365     lSingleSegmentPathPeriodFromDestination_iml_ptr->getDestination();
00366
00367     // Build the i-length SegmentPathPeriod
00368     // Get the parameters of the last segment
00369     stdair::SegmentPeriod* lSegmentPeriod_1_ptr =
00370     lSingleSegmentPathPeriodFromDestination_iml_ptr->getFirstSegmentPeriod(
);
00371     assert (lSegmentPeriod_1_ptr != NULL);
00372
00373     // Calculate the number of airlines flown by the i-length
00374     // segment path period
00375     const stdair::FlightPeriod& lFlightPeriod = stdair::BomManager::
00376     getParent<stdair::FlightPeriod> (*lSegmentPeriod_1_ptr);
00377     const stdair::Inventory& lInventory =
00378     stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00379     const stdair::AirlineCode_T& lAirlineCode_1 = lInventory.getAirlineCode();
00380
00381     stdair::NbOfAirlines_T lNbOfAirlines_i =
00382     lSegmentPathPeriod_iml_ptr->getNbOfAirlines();
00383     if (lSegmentPathPeriod_iml_ptr->isAirlineFlown(lAirlineCode_1) == false){
00384
00385         ++lNbOfAirlines_i;
00386     }
00387     lSegmentPathPeriodKey_i.setNbOfAirlines (lNbOfAirlines_i);
00388
00389     // Create the new segment path and add it to the dedicated lists.
00390     OriginDestinationSet* lOriginDestinationSet_ptr = stdair::BomManager::
00391     getObjectPtr<OriginDestinationSet>(ioReachableUniverse,lDestination_i);
00392
00393     if (lOriginDestinationSet_ptr == NULL) {
00394         OriginDestinationSetKey lKey (lDestination_i);
00395         lOriginDestinationSet_ptr =
00396         &stdair::FacBom<OriginDestinationSet>::instance().create (lKey);
00397         stdair::FacBomManager::addToListAndMap (ioReachableUniverse,
00398         *lOriginDestinationSet_ptr);
00399         stdair::FacBomManager::linkWithParent (ioReachableUniverse,
00400         *lOriginDestinationSet_ptr);
00401     }
00402     assert (lOriginDestinationSet_ptr != NULL);
00403
00404     SegmentPathPeriod& lSegmentPathPeriod_i = stdair::

```



```
00403         FacBom<SegmentPathPeriod>::instance().create (lSegmentPathPeriodKey_i);

00404         stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
00405                                           lSegmentPathPeriod_i);
00406         stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
00407                                                lSegmentPathPeriod_i);
00408
00409         // Clone the list of SegmentPeriod references of the given
00410         // SegmentPathPeriod object (passed as the second parameter).
00411         stdair::FacBomManager::
00412             cloneHolder<stdair::SegmentPeriod> (lSegmentPathPeriod_i,
00413                                                 *lSegmentPathPeriod_iml_ptr);
00414
00415
00416         // Add the SegmentPeriod reference to the dedicated list within
00417         // the SegmentPathPeriod. Note that this must be done before
00418         // the link between the SegmentPathPeriod and
00419         // ReachableUniverse, as that latter method uses the number of
00420         // segments within the SegmentPathPeriod object.
00421         stdair::FacBomManager::addToList (lSegmentPathPeriod_i,
00422                                           *lSegmentPeriod_l_ptr);
00423
00431         addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod_i);
00432     }
00433 }
00434 }
00435 }
```

## 26.113 airtsp/command/SegmentPathGenerator.hpp File Reference

```
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
```

### Classes

- class [AIRTSP::SegmentPathGenerator](#)  
*Class handling the generation / instantiation of the network BOM.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

**26.114 SegmentPathGenerator.hpp**

```

00001 #ifndef __AIRTSP_CMD_SEGMENTPATHGENERATOR_HPP
00002 #define __AIRTSP_CMD_SEGMENTPATHGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012 // AirTSP
00013 #include <airtsp/AIRTSP_Types.hpp>
00014
00015 namespace stdair {
00016     class BomRoot;
00017     class Inventory;
00018     class FlightPeriod;
00019     class SegmentPeriod;
00020 }
00021
00022 namespace AIRTSP {
00023     class ReachableUniverse;
00024     class OriginDestinationSet;
00025     class SegmentPathPeriod;
00026
00027     class SegmentPathGenerator : public stdair::CmdAbstract {
00028     public:
00029         static void createSegmentPathNetwork (const stdair::BomRoot&);
00030
00031     private:
00032         static void createSinglePaths (const stdair::Inventory&);
00033         static void createSinglePaths (const stdair::FlightPeriod&);
00034
00035         static void createSinglePath (stdair::SegmentPeriod&);
00036         static void createSinglePath (ReachableUniverse&, stdair::SegmentPeriod&);
00037
00038         static void buildSegmentPathNetwork (const stdair::BomRoot&,
00039                                             const stdair::NbOfSegments_T&);
00040         static void buildSegmentPathNetwork (ReachableUniverse&,
00041                                             const stdair::NbOfSegments_T&);
00042
00043         static void addSegmentPathPeriod (ReachableUniverse&,
00044                                           const SegmentPathPeriod&);
00045     };
00046 }
00047
00048 #endif // __AIRTSP_CMD_SEGMENTPATHGENERATOR_HPP

```

## 26.115 airtsp/command/SegmentPathProvider.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightPeriod.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/basic/BasConst_General.hpp>
#include <airtsp/bom/ReachableUniverse.hpp>
#include <airtsp/bom/OriginDestinationSet.hpp>
#include <airtsp/bom/SegmentPathPeriod.hpp>
#include <airtsp/command/SegmentPathProvider.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.116 SegmentPathProvider.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 // StdAir
00009 #include <stdair/basic/BasConst_BomDisplay.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/bom/Inventory.hpp>
00013 #include <stdair/bom/FlightPeriod.hpp>
00014 #include <stdair/bom/SegmentPeriod.hpp>
00015 #include <stdair/bom/BookingRequestStruct.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirTSP
00019 #include <airtsp/basic/BasConst_General.hpp>
00020 #include <airtsp/bom/ReachableUniverse.hpp>
00021 #include <airtsp/bom/OriginDestinationSet.hpp>
00022 #include <airtsp/bom/SegmentPathPeriod.hpp>
00023 #include <airtsp/command/SegmentPathProvider.hpp>
00024
00025 namespace AIRTSP {
00026
00027 // //////////////////////////////////////
00028 void SegmentPathProvider::
00029 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00030                      const stdair::BomRoot& iBomRoot,
00031                      const stdair::BookingRequestStruct& iBookingRequest) {
00032     // Retrieve the reachable universe object corresponding to the
00033     // origin of the booking request.
00034     const stdair::AirportCode_T& lOrigin = iBookingRequest.getOrigin ();
00035     const ReachableUniverse* lReachableUniverse_ptr =
00036         stdair::BomManager::getObjectPtr<ReachableUniverse> (iBomRoot, lOrigin);
00037     if (lReachableUniverse_ptr != NULL) {
00038         buildSegmentPathList (ioTravelSolutionList, *lReachableUniverse_ptr,
00039                             iBookingRequest);
00040     }
00041 }
00042
00043 // //////////////////////////////////////
00044 void SegmentPathProvider::
00045 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00046                      const ReachableUniverse& iReachableUniverse,
00047                      const stdair::BookingRequestStruct& iBookingRequest) {
00048     // Retrieve the origin-destination set objet corresponding to the
00049     // destination of the booking request.
00050     const stdair::AirportCode_T& lDestination = iBookingRequest.getDestination();
00051
00052     const OriginDestinationSet* lOriginDestinationSet_ptr =
00053         stdair::BomManager::getObjectPtr<OriginDestinationSet> (iReachableUniverse,
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999

```

```

00064         const stdair::BookingRequestStruct& iBookingRequest) {
00065     // Retrieve the departure date of the booking request.
00066     const stdair::Date_T& lPreferredDepartureDate =
00067         iBookingRequest.getPreferredDepartureDate ();
00068
00069     // Browse the list of segment path periods and find those which content
00070     // the preferred departure date.
00071     const SegmentPathPeriodList_T& lSegmentPathPeriodList =
00072         stdair::BomManager::getList<SegmentPathPeriod> (iOriginDestinationSet);
00073     for (SegmentPathPeriodList_T::const_iterator itSegmentPath =
00074         lSegmentPathPeriodList.begin ();
00075         itSegmentPath != lSegmentPathPeriodList.end (); ++itSegmentPath) {
00076         const SegmentPathPeriod* lCurrentSegmentPath_ptr = *itSegmentPath;
00077         assert (lCurrentSegmentPath_ptr != NULL);
00078         if (lCurrentSegmentPath_ptr->isDepartureDateValid(lPreferredDepartureDate)){
00079
00080             const stdair::DateTime_T lRequestDateTime =
00081                 iBookingRequest.getRequestDateTime();
00082             const stdair::Duration_T& lBoardingTime =
00083                 lCurrentSegmentPath_ptr->getBoardingTime();
00084             const stdair::DateTime_T lDepartureDateTime (lPreferredDepartureDate,
00085                                                         lBoardingTime);
00086             const bool IsDepartureDateValid =
00087                 ((lRequestDateTime + MINIMUM_TIME_BETWEEN_REQUEST_AND_DEPARTURE) <= lDe
00088                 partureDateTime);
00089             if (IsDepartureDateValid == false) {
00090                 return;
00091             }
00092             buildSegmentPathList (ioTravelSolutionList, *lCurrentSegmentPath_ptr,
00093                                 iBookingRequest);
00094         }
00095     }
00096
00097     void SegmentPathProvider::
00098     buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00099                          const SegmentPathPeriod& iSegmentPathPeriod,
00100                          const stdair::BookingRequestStruct& iBookingRequest) {
00101         // Create a new travel solution.
00102         stdair::TravelSolutionStruct lTravelSolution;
00103
00104         // Browse the list of segments and retrieve the necessary informations
00105         // for identifying the corresponding segment-date.
00106         const stdair::Date_T& lPreferredDepartureDate =
00107             iBookingRequest.getPreferredDepartureDate ();
00108         const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00109             stdair::BomManager::getList<stdair::SegmentPeriod> (iSegmentPathPeriod);
00110         const DateOffsetList_T& lBoardingDateOffsetList =
00111             iSegmentPathPeriod.getBoardingDateOffsetList ();
00112         assert (lSegmentPeriodList.size() == lBoardingDateOffsetList.size());
00113         DateOffsetList_T::const_iterator itOffset = lBoardingDateOffsetList.begin();
00114         for (stdair::SegmentPeriodList_T::const_iterator itSegment =
00115             lSegmentPeriodList.begin();
00116             itSegment != lSegmentPeriodList.end(); ++itSegment) {
00117             const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegment;
00118             assert (lSegmentPeriod_ptr != NULL);
00119             const stdair::DateOffset_T& lBoardingDateOffset = *itOffset;
00120
00121             // Find the corresponding segment-date within the segment period.
00122             const stdair::DateOffset_T& lSegmentBoardingDateOffset =
00123                 lSegmentPeriod_ptr->getBoardingDateOffset ();
00124             const stdair::Date_T& lReferenceFlightDate = lPreferredDepartureDate
00125                 + lBoardingDateOffset - lSegmentBoardingDateOffset;
00126
00127             // Build the whole segment-date key string.
00128             const stdair::FlightPeriod& lFlightPeriod =

```

```
00129         stdair::BomManager::getParent<stdair::FlightPeriod>(*lSegmentPeriod_ptr);

00130         const stdair::Inventory& lInventory =
00131             stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00132         const stdair::Duration_T lBoardingTime = lSegmentPeriod_ptr->getBoardingTime();
00133     e();
00133         std::ostringstream oStr;
00134         oStr << lInventory.getAirlineCode()
00135             << stdair::DEFAULT_KEY_FLD_DELIMITER
00136             << lFlightPeriod.getFlightNumber()
00137             << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00138             << boost::gregorian::to_simple_string (lReferenceFlightDate)
00139             << stdair::DEFAULT_KEY_FLD_DELIMITER
00140             << lSegmentPeriod_ptr->getBoardingPoint()
00141             << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00142             << lSegmentPeriod_ptr->getOffPoint()
00143             << stdair::DEFAULT_KEY_FLD_DELIMITER
00144             << lBoardingTime;
00145
00146         lTravelSolution.addSegment (oStr.str());
00147
00148         ++itOffset;
00149     }
00150     ioTravelSolutionList.push_back (lTravelSolution);
00151 }
00152
00153 }
```

## 26.117 airtsp/command/SegmentPathProvider.hpp File Reference

```
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

### Classes

- class [AIRTSP::SegmentPathProvider](#)  
*Class building the travel solutions from airline schedules.*

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)



**26.118 SegmentPathProvider.hpp**

```

00001 #ifndef __AIRTSP_COM_CMD_SEGMENTPATHPROVIDER_HPP
00002 #define __AIRTSP_COM_CMD_SEGMENTPATHPROVIDER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/TravelSolutionTypes.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00011 namespace stdair {
00012     class BomRoot;
00013     struct BookingRequestStruct;
00014 }
00015
00016 namespace AIRTSP {
00017     class ReachableUniverse;
00018     class OriginDestinationSet;
00019     class SegmentPathPeriod;
00020
00021     class SegmentPathProvider : public stdair::CmdAbstract {
00022     friend class AIRTSP_Service;
00023
00024     private:
00025         // ////////////////////////////////// Business Methods //////////////////////////////////
00026         static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00027                                           const stdair::BomRoot&,
00028                                           const stdair::BookingRequestStruct&);
00029
00030         static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00031                                           const ReachableUniverse&,
00032                                           const stdair::BookingRequestStruct&);
00033
00034         static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00035                                           const OriginDestinationSet&,
00036                                           const stdair::BookingRequestStruct&);
00037
00038         static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00039                                           const SegmentPathPeriod&,
00040                                           const stdair::BookingRequestStruct&);
00041     };
00042 }
00043
00044 #endif // __AIRTSP_COM_CMD_SEGMENTPATHPROVIDER_HPP

```

## 26.119 airtsp/command/Simulator.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/command/Simulator.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.120 Simulator.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 // StdAir
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/BookingRequestStruct.hpp>
00012 #include <stdair/service/Logger.hpp>
00013 // AirTSP
00014 #include <airtsp/command/Simulator.hpp>
00015
00016 namespace AIRTSP {
00017
00018 // //////////////////////////////////////
00019 void Simulator::simulate (stdair::BomRoot& ioBomRoot) {
00020
00021     // Delegate to the dedicated StdAir utility class
00022     // std::ostringstream oStream;
00023     // stdair::BomManager::display (oStream, ioBomRoot);
00024
00025     // DEBUG
00026     // STDAIR_LOG_DEBUG ("BOM Tree: ");
00027     // STDAIR_LOG_DEBUG (oStream.str());
00028
00029     // TODO: do not hardcode the booking request (get it from the
00030     // demand generation module instead).
00031     // stdair::BookingRequestStruct ("LHR", "JFK", stdair::Date_T (2009, 01, 16),
00032
00033     //                                     stdair::DEFAULT_DATETIME, "Y", 1);
00034 }
00035 }

```

## 26.121 airtsp/command/Simulator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

### Classes

- class [AIRTSP::Simulator](#)

### Namespaces

- namespace [stdair](#)  
*Forward declarations.*
- namespace [AIRTSP](#)

## 26.122 Simulator.hpp

```
00001 #ifndef __AIRTSP_COM_CMD_SIMULATOR_HPP
00002 #define __AIRTSP_COM_CMD_SIMULATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009
00010 // Forward declarations
00011 namespace stdair {
00012     class BomRoot;
00013 }
00014
00015 namespace AIRTSP {
00016
00018     class Simulator : public stdair::CmdAbstract {
00019     public:
00020
00021         // ////////// Business Methods //////////
00024         static void simulate (stdair::BomRoot&);
00025     };
00026
00027 }
00028 #endif // __AIRTSP_COM_CMD_SIMULATOR_HPP
```

## 26.123 airtsp/command/TravelSolutionParser.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <cassert>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airtsp/command/TravelSolutionParser.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.124 TravelSolutionParser.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <fstream>
00007 #include <cassert>
00008 // StdAir
00009 #include <stdair/stdair_exceptions.hpp>
00010 #include <stdair/basic/BasConst_TravelSolution.hpp>
00011 #include <stdair/basic/BasFileMgr.hpp>
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 // AirTSP
00015 #include <airtsp/command/TravelSolutionParser.hpp>
00016
00017 namespace AIRTSP {
00018
00019 // //////////////////////////////////////
00020 bool TravelSolutionParser::
00021 parseInputFileAndBuildBom (const std::string& iInputFileName) {
00022     bool hasReadBeenSuccessful = false;
00023
00024     // Check that the file path given as input corresponds to an actual file
00025     const bool doesExistAndIsReadable =
00026         stdair::BasFileMgr::doesExistAndIsReadable (iInputFileName);
00027     if (doesExistAndIsReadable == false) {
00028         std::ostringstream oMessage;
00029         oMessage << "The input file, '" << iInputFileName
00030             << "', can not be retrieved on the file-system";
00031         throw stdair::FileNotFoundException (oMessage.str());
00032     }
00033
00034     // Open the input file
00035     std::ifstream inputFile (iInputFileName.c_str());
00036     if (! inputFile) {
00037         STDAIR_LOG_ERROR ("Can not open input file '" << iInputFileName << "'");
00038         throw new stdair::FileNotFoundException ("Can not open input file '"
00039             + iInputFileName + "'");
00040     }
00041
00042     char buffer[80];
00043     double dval = 0.0;
00044     std::string dvalStr;
00045     short i = 1;
00046     bool hasAllParams = true;
00047
00048     stdair::AirportCode_T dAirport;
00049     stdair::AirportCode_T aAirport;
00050     stdair::Date_T depDate;
00051     stdair::Duration_T depTime;
00052     stdair::Duration_T arTime;
00053     stdair::Duration_T dur;
00054     //bool Ref;
00055     stdair::AirlineCode_T airline;
00056     stdair::CabinCode_T cabin;
00057     //stdair::FlightNumber_T flightNum;
00058     //stdair::Fare_T fare;
00059     //int lagsNum;
00060     //bool SNS;
00061     //bool change;
00062
00063     while (inputFile.getline (buffer, sizeof (buffer), ';')) {
00064         std::istringstream iStringStr (buffer);
00065

```

```
00066     bool hasRead = false;
00067
00068     if (i == 1) {
00069         hasAllParams = true;
00070     }
00071
00072     if (i>=1 && i<=14) {
00073         hasRead = (iStringStr >> dvalStr);
00074     }
00075
00076     if (i == 15) {
00077         hasRead = (iStringStr >> dval);
00078     }
00079
00080     if (hasRead) {
00081         if (i == 1) {
00082             dAirport = dvalStr;
00083
00084         } else if (i == 2) {
00085             aAirport = dvalStr;
00086             // std::cout << "City Pair = '" << dAiport
00087             // << "-" << aAirport << "'" << std::endl;
00088
00089         } else if (i == 3) {
00090             depDate = boost::gregorian::from_simple_string (dvalStr);
00091             // std::cout << "Date = '" << depDate << "'" << std::endl;
00092
00093         } else if (i == 4) {
00094             depTime = boost::posix_time::duration_from_string (dvalStr);
00095
00096         } else if (i == 5) {
00097             arTime = boost::posix_time::duration_from_string (dvalStr);
00098
00099         } else if (i == 6) {
00100             dur = boost::posix_time::duration_from_string (dvalStr);
00101
00102         } else if (i == 7) {
00103             //if (dvalStr == "refundable fare")
00104             //    Ref = true;
00105             //else Ref = false;
00106
00107         } else if (i == 8) {
00108             airline = dvalStr;
00109
00110         } else if (i == 9) {
00111             cabin = dvalStr;
00112
00113         } else if (i == 10) {
00114             //flightNum = dval;
00115
00116         } else if (i == 11) {
00117             //fare = dval;
00118
00119         } else if (i == 12) {
00120             //lagsNum = dval;
00121
00122         } else if (i == 13) {
00123             //if (dvalStr == "Saturday Nigth Stay mandatory")
00124             //    SNS = true;
00125             //else SNS = false;
00126
00127         } else if (i == 14) {
00128             //if (dvalStr == "changeable fare")
00129             //    change = true;
00130             //else change = false;
00131             i = 0;
00132         }
00133     }
```



```
00133
00134         //
00135         ++i;
00136
00137     } else {
00138         hasAllParams = false;
00139     }
00140 }
00141
00142 if (hasAllParams && i == 1) {
00143     STDAIR_LOG_DEBUG ("Successfully read");
00144 }
00145
00146 //
00147 if (!inputFile.eof()) {
00148     STDAIR_LOG_ERROR ("Problem when reading input file '" << iInputFileName
00149                     << "'");
00150     return hasReadBeenSuccessful;
00151 }
00152
00153 //
00154 hasReadBeenSuccessful = true;
00155 return hasReadBeenSuccessful;
00156 }
00157
00158 }
```

## 26.125 airtsp/command/TravelSolutionParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

### Classes

- class [AIRTSP::TravelSolutionParser](#)  
*Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file.*

### Namespaces

- namespace [AIRTSP](#)

**26.126 TravelSolutionParser.hpp**

```
00001 #ifndef __AIRTSP_CMD_TRAVELSOLUTIONPARSER_HPP
00002 #define __AIRTSP_CMD_TRAVELSOLUTIONPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012
00013 namespace AIRTSP {
00014
00019     class TravelSolutionParser : public stdair::CmdAbstract {
00020     public:
00028         static bool parseInputFileAndBuildBom (const stdair::Filename_T&);
00029     };
00030 }
00031 #endif // __AIRTSP_CMD_TRAVELSOLUTIONPARSER_HPP
```

## 26.127 airtsp/config/airtsp-paths.hpp.in File Reference

## 26.128 airtsp-paths.hpp.in

```
00001
00005 #ifndef __AIRTSP_PATHS_HPP__
00006 #define __AIRTSP_PATHS_HPP__
00007
00008 #define PACKAGE "@PACKAGE@"
00009 #define PACKAGE_NAME "@PACKAGE_NAME@"
00010 #define PACKAGE_VERSION "@PACKAGE_VERSION@"
00011 #define PREFIXDIR "@prefix@"
00012 #define EXEC_PREFIX "@exec_prefix@"
00013 #define BINDIR "@bindir@"
00014 #define LIBDIR "@libdir@"
00015 #define LIBEXECDIR "@libexecdir@"
00016 #define SBINDIR "@sbindir@"
00017 #define SYSCONFDIR "@sysconfdir@"
00018 #define INCLUDEDIR "@includedir@"
00019 #define DATAROOTDIR "@datarootdir@"
00020 #define DATADIR "@datadir@"
00021 #define DOCDIR "@docdir@"
00022 #define MANDIR "@mandir@"
00023 #define INFODIR "@infodir@"
00024 #define HTMLDIR "@htmldir@"
00025 #define PDFDIR "@pdfdir@"
00026 #define STDAIR_SAMPLE_DIR "@sampledir@"
00027
00028 #endif // __AIRTSP_PATHS_HPP__
00029
```

## 26.129 airtsp/factory/FacAIRTSPServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airtsp/factory/FacAIRTSPServiceContext.hpp>
#include <airtsp/service/AIRTSP_ServiceContext.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.130 FacAIRTSPServiceContext.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirTSP
00009 #include <airtsp/factory/FacAIRTSPServiceContext.hpp>
00010 #include <airtsp/service/AIRTSP_ServiceContext.hpp>
00011
00012 namespace AIRTSP {
00013
00014     FacAIRTSPServiceContext* FacAIRTSPServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacAIRTSPServiceContext::~FacAIRTSPServiceContext () {
00018         _instance = NULL;
00019     }
00020
00021     // //////////////////////////////////////
00022     FacAIRTSPServiceContext& FacAIRTSPServiceContext::instance () {
00023
00024         if (_instance == NULL) {
00025             _instance = new FacAIRTSPServiceContext ();
00026             assert (_instance != NULL);
00027
00028             stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029         }
00030         return *_instance;
00031     }
00032
00033     // //////////////////////////////////////
00034     AIRTSP_ServiceContext& FacAIRTSPServiceContext::create () {
00035         AIRTSP_ServiceContext* aServiceContext_ptr = NULL;
00036
00037         aServiceContext_ptr = new AIRTSP_ServiceContext ();
00038         assert (aServiceContext_ptr != NULL);
00039
00040         // The new object is added to the Bom pool
00041         _pool.push_back (aServiceContext_ptr);
00042
00043         return *aServiceContext_ptr;
00044     }
00045
00046 }

```

## 26.131 airtsp/factory/FacAIRTSPServiceContext.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
```

### Classes

- class [AIRTSP::FacAIRTSPServiceContext](#)  
*Factory for the service context.*

### Namespaces

- namespace [AIRTSP](#)



## 26.132 FacAIRTSPServiceContext.hpp

```
00001 #ifndef __AIRTSP_FAC_FACAIRTSPSERVICECONTEXT_HPP
00002 #define __AIRTSP_FAC_FACAIRTSPSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/service/FacServiceAbstract.hpp>
00010
00011 namespace AIRTSP {
00012
00013     class AIRTSP_ServiceContext;
00014
00015     class FacAIRTSPServiceContext : public stdair::FacServiceAbstract {
00016     public:
00017
00018         static FacAIRTSPServiceContext& instance();
00019
00020         ~FacAIRTSPServiceContext();
00021
00022         AIRTSP_ServiceContext& create();
00023
00024     protected:
00025         FacAIRTSPServiceContext() {}
00026
00027     private:
00028         static FacAIRTSPServiceContext* _instance;
00029
00030     };
00031 }
00032 #endif // __AIRTSP_FAC_FACAIRTSPSERVICECONTEXT_HPP
```

## 26.133 airtsp/factory/FacServiceAbstract.cpp File Reference

```
#include <cassert>
#include <airtsp/service/ServiceAbstract.hpp>
#include <airtsp/factory/FacServiceAbstract.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.134 FacServiceAbstract.cpp**

```
00001 // ////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AirTSP
00007 #include <airtsp/service/ServiceAbstract.hpp>
00008 #include <airtsp/factory/FacServiceAbstract.hpp>
00009
00010 namespace AIRTSP {
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 }
```

## 26.135 airtsp/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

### Classes

- class [AIRTSP::FacServiceAbstract](#)

### Namespaces

- namespace [AIRTSP](#)

## 26.136 FacServiceAbstract.hpp

```
00001 #ifndef __AIRTSP_FAC_FACSERVICEABSTRACT_HPP
00002 #define __AIRTSP_FAC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRTSP {
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 #endif // __AIRTSP_FAC_FACSERVICEABSTRACT_HPP
```

## 26.137 airtsp/service/AIRTSP\_Service.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
#include <airtsp/factory/FacAIRTSPServiceContext.hpp>
#include <airtsp/command/Simulator.hpp>
#include <airtsp/command/ScheduleParser.hpp>
#include <airtsp/command/OnDParser.hpp>
#include <airtsp/command/SegmentPathProvider.hpp>
#include <airtsp/command/InventoryGenerator.hpp>
#include <airtsp/command/SegmentPathGenerator.hpp>
#include <airtsp/service/AIRTSP_ServiceContext.hpp>
#include <airtsp/AIRTSP_Service.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.138 AIRTSP\_Service.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/basic/BasChronometer.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BookingRequestStruct.hpp>
00013 #include <stdair/bom/TravelSolutionStruct.hpp>
00014 #include <stdair/service/Logger.hpp>
00015 #include <stdair/STDAIR_Service.hpp>
00016 // AirtSP
00017 #include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
00018 #include <airtsp/factory/FacAIRTSPServiceContext.hpp>
00019 #include <airtsp/command/Simulator.hpp>
00020 #include <airtsp/command/ScheduleParser.hpp>
00021 #include <airtsp/command/OnDParser.hpp>
00022 #include <airtsp/command/SegmentPathProvider.hpp>
00023 #include <airtsp/command/InventoryGenerator.hpp>
00024 #include <airtsp/command/SegmentPathGenerator.hpp>
00025 #include <airtsp/service/AIRTSP_ServiceContext.hpp>
00026 #include <airtsp/AIRTSP_Service.hpp>
00027
00028 namespace AIRTSP {
00029
00030 // //////////////////////////////////////
00031 AIRTSP_Service::AIRTSP_Service() : _airtspServiceContext (NULL) {
00032     assert (false);
00033 }
00034
00035 // //////////////////////////////////////
00036 AIRTSP_Service::AIRTSP_Service (const AIRTSP_Service& iService)
00037 : _airtspServiceContext (NULL) {
00038     assert (false);
00039 }
00040
00041 // //////////////////////////////////////
00042 AIRTSP_Service::AIRTSP_Service (const stdair::BasLogParams& iLogParams)
00043 : _airtspServiceContext (NULL) {
00044
00045     // Initialise the STDAIR service handler
00046     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00047         initStdAirService (iLogParams);
00048
00049     // Initialise the service context
00050     initServiceContext();
00051
00052     // Add the StdAir service context to the Airtsp service context
00053     // \note Airtsp owns the STDAIR service resources here.
00054     const bool ownStdairService = true;
00055     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00056
00057     // Initialise the (remaining of the) context
00058     initAirtspService();
00059 }
00060
00061 // //////////////////////////////////////
00062 AIRTSP_Service::AIRTSP_Service (const stdair::BasLogParams& iLogParams,
00063                                 const stdair::BasDBParams& iDBParams)
00064 : _airtspServiceContext (NULL) {
00065

```

```

00066     // Initialise the STDAIR service handler
00067     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00068         initStdAirService (iLogParams, iDBParams);
00069
00070     // Initialise the service context
00071     initServiceContext();
00072
00073     // Add the StdAir service context to the Airtsp service context
00074     // \note Airtsp owns the STDAIR service resources here.
00075     const bool ownStdairService = true;
00076     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00077
00078     // Initialise the (remaining of the) context
00079     initAirtspService();
00080 }
00081
00082 // //////////////////////////////////////
00083 AIRTSP_Service::
00084 AIRTSP_Service (stdair::STDAIR_ServicePtr_T ioSTDAIRServicePtr)
00085     : _airtspServiceContext (NULL) {
00086
00087     // Initialise the service context
00088     initServiceContext();
00089
00090     // Add the StdAir service context to the Airtsp service context.
00091     // \note Airtsp does not own the STDAIR service resources here.
00092     const bool doesNotOwnStdairService = false;
00093     addStdAirService (ioSTDAIRServicePtr, doesNotOwnStdairService);
00094
00095     // Initialise the context
00096     initAirtspService();
00097 }
00098
00099 // //////////////////////////////////////
00100 AIRTSP_Service::~AIRTSP_Service() {
00101     // Delete/Clean all the objects from memory
00102     finalise();
00103 }
00104
00105 // //////////////////////////////////////
00106 void AIRTSP_Service::finalise() {
00107     assert (_airtspServiceContext != NULL);
00108     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00109     _airtspServiceContext->reset();
00110 }
00111
00112 // //////////////////////////////////////
00113 void AIRTSP_Service::initServiceContext() {
00114     // Initialise the service context
00115     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00116         FacAIRTSPServiceContext::instance().create();
00117     _airtspServiceContext = &lAIRTSP_ServiceContext;
00118 }
00119
00120 // //////////////////////////////////////
00121 void AIRTSP_Service::
00122 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00123                 const bool iOwnStdairService) {
00124
00125     // Retrieve the Airtsp service context
00126     assert (_airtspServiceContext != NULL);
00127     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00128         *_airtspServiceContext;
00129
00130     // Store the STDAIR service object within the (Airtsp) service context
00131     lAIRTSP_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00132                                             iOwnStdairService);

```



```

00133     }
00134
00135     // ////////////////////////////////////////
00136     stdair::STDAIR_ServicePtr_T AIRTSP_Service::
00137     initStdAirService (const stdair::BasLogParams& iLogParams) {
00138
00139         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00140             boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00141
00142         return lSTDAIR_Service_ptr;
00143     }
00144
00145     // ////////////////////////////////////////
00146     stdair::STDAIR_ServicePtr_T AIRTSP_Service::
00147     initStdAirService (const stdair::BasLogParams& iLogParams,
00148                       const stdair::BasDBParams& iDBParams) {
00149
00150         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00151             boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00152
00153         return lSTDAIR_Service_ptr;
00154     }
00155
00156     // ////////////////////////////////////////
00157     void AIRTSP_Service::initAirtspService() {
00158         // Do nothing at this stage. A sample BOM tree may be built by
00159         // calling the buildSampleBom() method
00160     }
00161
00162     // ////////////////////////////////////////
00163     void AIRTSP_Service::
00164     parseAndLoad (const stdair::ScheduleFilePath& iScheduleInputFilePath) {
00165
00166         // Retrieve the BOM tree root
00167         assert (_airtspServiceContext != NULL);
00168         AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00169             *_airtspServiceContext;
00170         const bool doesOwnStdairService =
00171             lAIRTSP_ServiceContext.getOwnStdairServiceFlag();
00172
00173         // Retrieve the StdAir service object from the (Airtsp) service context
00174         stdair::STDAIR_Service& lSTDAIR_Service =
00175             lAIRTSP_ServiceContext.getSTDAIR_Service();
00176         stdair::BomRoot& lPersistentBomRoot =
00177             lSTDAIR_Service.getPersistentBomRoot();
00178
00179         stdair::BasChronometer lINVGeneration; lINVGeneration.start();
00180         ScheduleParser::generateInventories (iScheduleInputFilePath,
00181                                             lPersistentBomRoot);
00182         buildComplementaryLinks (lPersistentBomRoot);
00183
00184         const double lGenerationMeasure = lINVGeneration.elapsed();
00185
00186         if (doesOwnStdairService == true) {
00187             //
00188             clonePersistentBom ();
00189         }
00190
00191         // DEBUG
00192         STDAIR_LOG_DEBUG ("Inventory generation time: " << lGenerationMeasure);
00193     }
00194
00195     // ////////////////////////////////////////
00196     void AIRTSP_Service::
00197     parseAndLoad (const stdair::ScheduleFilePath& iScheduleInputFilePath,
00198                  const stdair::ODFilePath& iODInputFilePath) {

```

```

00232
00233     // First, build the airline inventories from the schedule file
00234     parseAndLoad (iScheduleInputFilePath);
00235
00236     // Retrieve the BOM tree root
00237     assert (_airtspServiceContext != NULL);
00238     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00239         *_airtspServiceContext;
00240     const bool doesOwnStdairService =
00241         lAIRTSP_ServiceContext.getOwnStdairServiceFlag();
00242
00243     // Retrieve the StdAir service object from the (Airtsp) service context
00244     stdair::STDAIR_Service& lSTDAIR_Service =
00245         lAIRTSP_ServiceContext.getSTDAIR_Service();
00246     stdair::BomRoot& lPersistentBomRoot =
00247         lSTDAIR_Service.getPersistentBomRoot();
00248
00252     stdair::BasChronometer lOnDGeneration; lOnDGeneration.start();
00253     OnDParser::generateOnDPeriods (iODInputFilePath, lPersistentBomRoot);
00254     const double lGenerationMeasure = lOnDGeneration.elapsed();
00255
00268     if (doesOwnStdairService == true) {
00269
00270         //
00271         lSTDAIR_Service.clonePersistentBom ();
00272     }
00273
00278     stdair::BomRoot& lBomRoot =
00279         lSTDAIR_Service.getBomRoot();
00280     buildComplementaryLinks (lBomRoot);
00281
00282     // DEBUG
00283     STDAIR_LOG_DEBUG ("O&D generation time: " << lGenerationMeasure);
00284 }
00285
00286 // =====
00287 void AIRTSP_Service::buildSampleBom() {
00288
00289     // Retrieve the Airtsp service context
00290     if (_airtspServiceContext == NULL) {
00291         throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00292             "not been initialised");
00293     }
00294     assert (_airtspServiceContext != NULL);
00295
00296     // Retrieve the Airtsp service context and whether it owns the Stdair
00297     // service
00298     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00299         *_airtspServiceContext;
00300     const bool doesOwnStdairService =
00301         lAIRTSP_ServiceContext.getOwnStdairServiceFlag();
00302
00303     // Retrieve the StdAir service object from the (Airtsp) service context
00304     stdair::STDAIR_Service& lSTDAIR_Service =
00305         lAIRTSP_ServiceContext.getSTDAIR_Service();
00306
00311     if (doesOwnStdairService == true) {
00312         //
00313         lSTDAIR_Service.buildSampleBom();
00314     }
00315
00328     stdair::BomRoot& lPersistentBomRoot =
00329         lSTDAIR_Service.getPersistentBomRoot();
00330     buildComplementaryLinks (lPersistentBomRoot);
00331
00336     if (doesOwnStdairService == true) {
00337

```

```

00338         //
00339         clonePersistentBom ();
00340     }
00341 }
00342
00343 // //////////////////////////////////////
00344 void AIRTSP_Service::clonePersistentBom () {
00345     // Retrieve the Airtsp service context
00346     if (_airtspServiceContext == NULL) {
00347         throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00348                                                     "not been initialised");
00349     }
00350     assert (_airtspServiceContext != NULL);
00351
00352     // Retrieve the Airtsp service context and whether it owns the Stdair
00353     // service
00354     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00355         *_airtspServiceContext;
00356     const bool doesOwnStdairService =
00357         lAIRTSP_ServiceContext.getOwnStdairServiceFlag();
00358
00359     // Retrieve the StdAir service object from the (Airtsp) service context
00360     stdair::STDAIR_Service& lSTDAIR_Service =
00361         lAIRTSP_ServiceContext.getSTDAIR_Service();
00362
00363     if (doesOwnStdairService == true) {
00364         //
00365         lSTDAIR_Service.clonePersistentBom ();
00366     }
00367
00368     stdair::BomRoot& lBomRoot =
00369         lSTDAIR_Service.getBomRoot();
00370     buildComplementaryLinks (lBomRoot);
00371 }
00372
00373 // //////////////////////////////////////
00374 void AIRTSP_Service::buildComplementaryLinks (stdair::BomRoot& ioBomRoot) {
00375     // Retrieve the Airtsp service context
00376     if (_airtspServiceContext == NULL) {
00377         throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00378                                                     "not been initialised");
00379     }
00380     assert (_airtspServiceContext != NULL);
00381
00382     SegmentPathGenerator::createSegmentPathNetwork (ioBomRoot);
00383 }
00384
00385 // //////////////////////////////////////
00386 std::string AIRTSP_Service::
00387 jsonExportFlightDateObjects (const stdair::AirlineCode_T& iAirlineCode,
00388                             const stdair::FlightNumber_T& iFlightNumber,
00389                             const stdair::Date_T& iDepartureDate) const {
00390     // Retrieve the Airtsp service context
00391     if (_airtspServiceContext == NULL) {
00392         throw stdair::NonInitialisedServiceException ("The Airtsp service "
00393                                                     "has not been initialised");
00394     }
00395     assert (_airtspServiceContext != NULL);
00396
00397     // Retrieve the StdAir service object from the (Airtsp) service context
00398     AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00399         *_airtspServiceContext;
00400     stdair::STDAIR_Service& lSTDAIR_Service =

```

```

00416         lAIRTSP_ServiceContext.getSTDAIR_Service();
00417
00418         // Delegate the JSON export to the dedicated service
00419         return lSTDAIR_Service.jsonExportFlightDateObjects (iAirlineCode,
00420                                                             iFlightNumber,
00421                                                             iDepartureDate);
00422     }
00423
00424     // //////////////////////////////////////
00425     std::string AIRTSP_Service::csvDisplay() const {
00426
00427         // Retrieve the Airtsp service context
00428         if (_airtspServiceContext == NULL) {
00429             throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00430                                                         "not been initialised");
00431         }
00432         assert (_airtspServiceContext != NULL);
00433
00434         // Retrieve the STDAIR service object from the (Airtsp) service context
00435         AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00436             *_airtspServiceContext;
00437         stdair::STDAIR_Service& lSTDAIR_Service =
00438             lAIRTSP_ServiceContext.getSTDAIR_Service();
00439         const stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00440
00441         // Delegate the BOM building to the dedicated service
00442         return lSTDAIR_Service.csvDisplay(lBomRoot);
00443     }
00444
00445     // //////////////////////////////////////
00446     std::string AIRTSP_Service::
00447     csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00448                const stdair::FlightNumber_T& iFlightNumber,
00449                const stdair::Date_T& iDepartureDate) const {
00450
00451         // Retrieve the Airtsp service context
00452         if (_airtspServiceContext == NULL) {
00453             throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00454                                                         "not been initialised");
00455         }
00456         assert (_airtspServiceContext != NULL);
00457
00458         // Retrieve the STDAIR service object from the (Airtsp) service context
00459         AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00460             *_airtspServiceContext;
00461         stdair::STDAIR_Service& lSTDAIR_Service =
00462             lAIRTSP_ServiceContext.getSTDAIR_Service();
00463
00464         // Delegate the BOM display to the dedicated service
00465         return lSTDAIR_Service.csvDisplay (iAirlineCode, iFlightNumber,
00466                                             iDepartureDate);
00467     }
00468
00469     // //////////////////////////////////////
00470     void AIRTSP_Service::simulate() {
00471
00472         // Retrieve the Airtsp service context
00473         if (_airtspServiceContext == NULL) {
00474             throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00475                                                         "not been initialised");
00476         }
00477         assert (_airtspServiceContext != NULL);
00478
00479         // Retrieve the BOM tree root
00480         AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00481             *_airtspServiceContext;
00482         stdair::STDAIR_Service& lSTDAIR_Service =

```

```

00483         lAIRTSP_ServiceContext.getSTDAIR_Service();
00484         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00485
00486         // Call the underlying Use Case (command)
00487         stdair::BasChronometer lSimulateChronometer; lSimulateChronometer.start();
00488         Simulator::simulate (lBomRoot);
00489         const double lSimulateMeasure = lSimulateChronometer.elapsed();
00490
00491         // DEBUG
00492         STDAIR_LOG_DEBUG ("Simulation: " << lSimulateMeasure << " - "
00493             << lAIRTSP_ServiceContext.display());
00494     }
00495
00496     // //////////////////////////////////////
00497     void AIRTSP_Service::
00498     buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00499         const stdair::BookingRequestStruct& iBookingRequest) {
00500
00501         if (_airtspServiceContext == NULL) {
00502             throw stdair::NonInitialisedServiceException ("The Airtsp service has "
00503                 "not been initialised");
00504         }
00505         assert (_airtspServiceContext != NULL);
00506
00507         // Retrieve the BOM tree root
00508         AIRTSP_ServiceContext& lAIRTSP_ServiceContext =
00509             *_airtspServiceContext;
00510         stdair::STDAIR_Service& lSTDAIR_Service =
00511             lAIRTSP_ServiceContext.getSTDAIR_Service();
00512         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00513
00514         // Delegate the call to the dedicated command
00515         stdair::BasChronometer lBuildChronometer; lBuildChronometer.start();
00516         SegmentPathProvider::buildSegmentPathList (ioTravelSolutionList,
00517             lBomRoot, iBookingRequest);
00518         const double lBuildMeasure = lBuildChronometer.elapsed();
00519
00520         // DEBUG
00521         STDAIR_LOG_DEBUG ("Segment-path build: " << lBuildMeasure << " - "
00522             << lAIRTSP_ServiceContext.display());
00523     }
00524
00525 }

```

## 26.139 airtsp/service/AIRTSP\_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/STDAIR_Service.hpp>
#include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
#include <airtsp/service/AIRTSP_ServiceContext.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.140 AIRTSP\_ServiceContext.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/STDAIR_Service.hpp>
00009 // AirTSP
00010 #include <airtsp/basic/BasConst_AIRTSP_Service.hpp>
00011 #include <airtsp/service/AIRTSP_ServiceContext.hpp>
00012
00013 namespace AIRTSP {
00014
00015     // //////////////////////////////////////
00016     AIRTSP_ServiceContext::AIRTSP_ServiceContext()
00017         : _ownStdairService (false) {
00018     }
00019
00020     // //////////////////////////////////////
00021     AIRTSP_ServiceContext::
00022     AIRTSP_ServiceContext (const AIRTSP_ServiceContext&) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     AIRTSP_ServiceContext::~AIRTSP_ServiceContext() {
00028     }
00029
00030     // //////////////////////////////////////
00031     stdair::STDAIR_Service& AIRTSP_ServiceContext::getSTDAIR_Service() const {
00032         assert (_stdairService != NULL);
00033         return *_stdairService;
00034     }
00035
00036     // //////////////////////////////////////
00037     const std::string AIRTSP_ServiceContext::shortDisplay() const {
00038         std::ostringstream ostr;
00039         ostr << "AIRTSP_ServiceContext -- Owns StdAir service: "
00040             << _ownStdairService;
00041         return ostr.str();
00042     }
00043
00044     // //////////////////////////////////////
00045     const std::string AIRTSP_ServiceContext::display() const {
00046         std::ostringstream ostr;
00047         ostr << shortDisplay();
00048         return ostr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     const std::string AIRTSP_ServiceContext::describe() const {
00053         return shortDisplay();
00054     }
00055
00056     // //////////////////////////////////////
00057     void AIRTSP_ServiceContext::reset() {
00058
00059         // The shared_ptr<>::reset() method drops the refcount by one.
00060         // If the count result is dropping to zero, the resource pointed to
00061         // by the shared_ptr<> will be freed.
00062
00063         // Reset the stdair shared pointer
00064         _stdairService.reset();
00065     }

```

```
00066  
00067 }
```



## 26.141 airtsp/service/AIRTSP\_ServiceContext.hpp File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <airtsp/AIRTSP_Types.hpp>
```

### Classes

- class [AIRTSP::AIRTSP\\_ServiceContext](#)  
*Class holding the context of the Airtsp services.*

### Namespaces

- namespace [AIRTSP](#)

**26.142 AIRTSP\_ServiceContext.hpp**

```

00001 #ifndef __AIRTSP_SVC_AIRTSP_SERVICE_CONTEXT_HPP
00002 #define __AIRTSP_SVC_AIRTSP_SERVICE_CONTEXT_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 // AirTSP
00015 #include <airtsp/AIRTSP_Types.hpp>
00016
00017 namespace AIRTSP {
00018
00022     class AIRTSP_ServiceContext : public stdair::ServiceAbstract {
00028         friend class AIRTSP_Service;
00029         friend class FacAIRTSPServiceContext;
00030
00031     private:
00032         // ////////////////////////////////// Getters //////////////////////////////////
00036         stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00037             return _stdairService;
00038         }
00039
00043         stdair::STDAIR_Service& getSTDAIR_Service() const;
00044
00048         const bool getOwnStdairServiceFlag() const {
00049             return _ownStdairService;
00050         }
00051
00052     private:
00053         // ////////////////////////////////// Setters //////////////////////////////////
00054         void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00055                                 const bool iOwnStdairService) {
00056             _stdairService = ioSTDAIR_ServicePtr;
00061             _ownStdairService = iOwnStdairService;
00062         }
00063
00064     private:
00066         // ////////////////////////////////// Display Methods //////////////////////////////////
00070         const std::string shortDisplay() const;
00071
00075         const std::string display() const;
00076
00080         const std::string describe() const;
00081
00082     private:
00083         AIRTSP_ServiceContext();
00088         AIRTSP_ServiceContext (const AIRTSP_ServiceContext&);
00093
00098         void init();
00099
00103         ~AIRTSP_ServiceContext();
00104
00108         void reset();
00109

```

```
00110
00111     private:
00112         // ////////////////////////////////// Children //////////////////////////////////
00116         stdair::STDAIR_ServicePtr_T _stdairService;
00117
00121         bool _ownStdairService;
00122     };
00123
00124 }
00125 #endif // __AIRTSP_SVC_AIRTSP_SERVICE_CONTEXT_HPP
```

## 26.143 airtsp/service/ServiceAbstract.cpp File Reference

```
#include <airtsp/service/ServiceAbstract.hpp>
```

### Namespaces

- namespace [AIRTSP](#)

**26.144 ServiceAbstract.cpp**

```
00001 // ////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////
00004 // AIRTSP
00005 #include <airtsp/service/ServiceAbstract.hpp>
00006
00007 namespace AIRTSP {
00008
00009 }
```

## 26.145 airtsp/service/ServiceAbstract.hpp File Reference

```
#include <iostream>
#include <sstream>
```

### Classes

- class [AIRTSP::ServiceAbstract](#)

### Namespaces

- namespace [AIRTSP](#)

### Functions

- `template<class charT , class traits > std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const AIRTSP::ServiceAbstract &iService)`
- `template<class charT , class traits > std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, AIRTSP::ServiceAbstract &ioService)`

#### 26.145.1 Function Documentation

**26.145.1.1** `template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & ioOut, const AIRTSP::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](#).

**26.145.1.2** `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & ioIn, AIRTSP::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 [AIRTSP::ServiceAbstract::fromStream\(\)](#).

**26.146 ServiceAbstract.hpp**

```

00001 #ifndef __AIRTSP_SERVICEABSTRACT_HPP
00002 #define __AIRTSP_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iostream>
00009 #include <sstream>
00010
00011 namespace AIRTSP {
00012
00013     class ServiceAbstract {
00014     public:
00015
00016         virtual ~ServiceAbstract() {}
00017
00018         virtual void toStream (std::ostream& ioOut) const {}
00019
00020         virtual void fromStream (std::istream& ioIn) {}
00021
00022     protected:
00023         ServiceAbstract() {}
00024     };
00025 }
00026
00027 template <class charT, class traits>
00028 inline
00029 std::basic_ostream<charT, traits>&
00030 operator<< (std::basic_ostream<charT, traits>& ioOut,
00031            const AIRTSP::ServiceAbstract& iService) {
00032     std::basic_ostringstream<charT,traits> ostr;
00033     ostr.copyfmt (ioOut);
00034     ostr.width (0);
00035
00036     // Fill string stream
00037     iService.toStream (ostr);
00038
00039     // Print string stream
00040     ioOut << ostr.str();
00041
00042     return ioOut;
00043 }
00044
00045 template <class charT, class traits>
00046 inline
00047 std::basic_istream<charT, traits>&
00048 operator>> (std::basic_istream<charT, traits>& ioIn,
00049            AIRTSP::ServiceAbstract& ioService) {
00050     // Fill Service object with input stream
00051     ioService.fromStream (ioIn);
00052     return ioIn;
00053 }
00054
00055 #endif // __AIRTSP_SERVICEABSTRACT_HPP

```

- 
- 26.147 doc/local/authors.doc File Reference
  - 26.148 doc/local/codingrules.doc File Reference
  - 26.149 doc/local/copyright.doc File Reference
  - 26.150 doc/local/documentation.doc File Reference
  - 26.151 doc/local/features.doc File Reference
  - 26.152 doc/local/help\_wanted.doc File Reference
  - 26.153 doc/local/howto\_release.doc File Reference
  - 26.154 doc/local/index.doc File Reference
  - 26.155 doc/local/installation.doc File Reference
  - 26.156 doc/local/linking.doc File Reference
  - 26.157 doc/local/test.doc File Reference
  - 26.158 doc/local/users\_guide.doc File Reference
  - 26.159 doc/local/verification.doc File Reference
  - 26.160 doc/tutorial/tutorial.doc File Reference
  - 26.161 test/airtsp/AirlineScheduleTestSuite.cpp File Reference



**26.162 AirlineScheduleTestSuite.cpp**

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost Unit Test Framework (UTF)
00013 #define BOOST_TEST_DYN_LINK
00014 #define BOOST_TEST_MAIN
00015 #define BOOST_TEST_MODULE InventoryTestSuite
00016 #include <boost/test/unit_test.hpp>
00017 // Boost Date-Time
00018 #include <boost/date_time/gregorian/gregorian.hpp>
00019 // StdAir
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/basic/BasLogParams.hpp>
00022 #include <stdair/basic/BasDBParams.hpp>
00023 #include <stdair/basic/BasFileMgr.hpp>
00024 #include <stdair/bom/TravelSolutionStruct.hpp>
00025 #include <stdair/bom/BookingRequestStruct.hpp>
00026 #include <stdair/service/Logger.hpp>
00027 // AirTSP
00028 #include <airtsp/AIRTSP_Types.hpp>
00029 #include <airtsp/AIRTSP_Service.hpp>
00030 #include <airtsp/config/airtsp-paths.hpp>
00031
00032 namespace boost_utf = boost::unit_test;
00033
00034 // (Boost) Unit Test XML Report
00035 std::ofstream utfReportStream ("AirlineScheduleTestSuite_utfresults.xml");
00036
00040 struct UnitTestConfig {
00042     UnitTestConfig() {
00043         boost_utf::unit_test_log.set_stream (utfReportStream);
00044         boost_utf::unit_test_log.set_format (boost_utf::XML);
00045         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00046         //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
00047     ts);
00048     }
00049
00050     ~UnitTestConfig() {
00051     }
00052 };
00053
00054 // //////////////////////////////////////
00058 const unsigned int testScheduleHelper (const unsigned short iTestFlag,
00059                                         const stdair::Filename_T& iScheduleInputF
00060                                         ilename,
00061                                         const stdair::Filename_T& iODInputFilenam
00062                                         e,
00063                                         const bool isBuiltin,
00064                                         const bool isWithOnD) {
00065
00066     // Output log File
00067     std::ostringstream oStr;
00068     oStr << "AirlineScheduleTestSuite_" << iTestFlag << ".log";
00069     const stdair::Filename_T lLogFilename (oStr.str());
00070
00071     // Set the log parameters
00072     std::ofstream logOutputFile;
00073     // Open and clean the log outputfile
00074     logOutputFile.open (lLogFilename.c_str());
00075     logOutputFile.clear();

```

```

00074
00075 // Instantiate the AirTSP service
00076 const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00077 AIRTSP::AIRTSP_Service airtspService (lLogParams);
00078
00079 stdair::AirportCode_T lOrigin;
00080 stdair::AirportCode_T lDestination;
00081 stdair::AirportCode_T lPOS;
00082 stdair::Date_T lPreferredDepartureDate;;
00083 stdair::Date_T lRequestDate;
00084
00085 // Check wether or not a (CSV) input file should be read
00086 if (isBuiltin == true) {
00087
00088     // Build the default sample BOM tree (filled with schedules)
00089     airtspService.buildSampleBom();
00090
00091     lOrigin = "SIN";
00092     lDestination = "BKK";
00093     lPOS = "SIN";
00094     lPreferredDepartureDate = boost::gregorian::from_string ("2010/02/08");
00095     lRequestDate = boost::gregorian::from_string ("2010/01/21");
00096
00097 } else {
00098
00099     if (isWithOnD == false) {
00100
00101         // Build the BOM tree from parsing input files
00102         const stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
00103         airtspService.parseAndLoad (lScheduleFilePath);
00104
00105         lOrigin = "NCE";
00106         lDestination = "BKK";
00107         lPOS = "NCE";
00108         lPreferredDepartureDate = boost::gregorian::from_string ("2007/04/21");
00109         lRequestDate = boost::gregorian::from_string ("2007/03/21");
00110
00111     } else {
00112
00113         // Build the BOM tree from parsing input files
00114         const stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
00115         const stdair::ODFilePath lODFilePath (iODInputFilename);
00116         airtspService.parseAndLoad (lScheduleFilePath,
00117                                     lODFilePath);
00118
00119         lOrigin = "SIN";
00120         lDestination = "BKK";
00121         lPOS = "SIN";
00122         lPreferredDepartureDate = boost::gregorian::from_string ("2012/06/04");
00123         lRequestDate = boost::gregorian::from_string ("2012/01/01");
00124     }
00125
00126 }
00127
00128 // Create a booking request structure
00129 const stdair::Duration_T lRequestTime (boost::posix_time::hours(8));
00130 const stdair::DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
00131 const stdair::CabinCode_T lPreferredCabin ("Bus");
00132 const stdair::PartySize_T lPartySize (3);
00133 const stdair::ChannelLabel_T lChannel ("DF");
00134 const stdair::TripType_T lTripType ("RO");
00135 const stdair::DayDuration_T lStayDuration (5);
00136 const stdair::FrequentFlyer_T lFrequentFlyerType ("NONE");
00137 const stdair::Duration_T lPreferredDepartureTime (boost::posix_time::hours(10))
00138 ;
00139 const stdair::WTP_T lWTP (2000.0);
00139 const stdair::PriceValue_T lValueOfTime (20.0);

```

```

00140     const stdair::ChangeFees_T lChangeFees (true);
00141     const stdair::Disutility_T lChangeFeeDisutility (50);
00142     const stdair::NonRefundable_T lNonRefundable (true);
00143     const stdair::Disutility_T lNonRefundableDisutility (50);
00144
00145     const stdair::BookingRequestStruct lBookingRequest (lOrigin, lDestination,
00146                                                         lPOS,
00147                                                         lPreferredDepartureDate,
00148                                                         lRequestDateTime,
00149                                                         lPreferredCabin,
00150                                                         lPartySize, lChannel,
00151                                                         lTripType, lStayDuration,
00152                                                         lFrequentFlyerType,
00153                                                         lPreferredDepartureTime,
00154                                                         lWTP, lValueOfTime,
00155                                                         lChangeFees,
00156                                                         lChangeFeeDisutility,
00157                                                         lNonRefundable,
00158                                                         lNonRefundableDisutility);
00159
00160     // Build the segment path list
00161     stdair::TravelSolutionList_T lTravelSolutionList;
00162     airtspService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
00163     const unsigned int lNbOfTravelSolutions = lTravelSolutionList.size();
00164
00165     STDAIR_LOG_DEBUG ("The number of travel solutions for the booking request '"
00166                     << lBookingRequest.describe() << "' is equal to "
00167                     << lNbOfTravelSolutions << ".");
00168
00169     // Close the Log outputFile
00170     logOutputFile.close();
00171
00172     return lNbOfTravelSolutions;
00173 }
00174
00175
00176 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00177
00178 // Set the UTF configuration (re-direct the output to a specific file)
00179 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00180
00181 // Start the test suite
00182 BOOST_AUTO_TEST_SUITE (master_test_suite)
00183
00184
00185 BOOST_AUTO_TEST_CASE (airtsp_simple_build) {
00186
00187     // Input file name
00188     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00189                                                         "/schedule03.csv");
00190
00191     // State whether the BOM tree should be built-in or parsed from input files
00192     const bool isBuiltin = false;
00193     const bool isWithOnD = false;
00194
00195     // Try to build a travel solution list
00196     unsigned int lNbOfTravelSolutions = 0;
00197     BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
00198                           testScheduleHelper (0, lScheduleInputFilename, " ",
00199                                               isBuiltin, isWithOnD));
00200
00201     // Check the size of the travel solution list
00202     const unsigned int lExpectedNbOfTravelSolutions = 4;
00203     BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
00204                        "The number of travel solutions is "
00205                        << lNbOfTravelSolutions << ", but it should be equal to "
00206                        << lExpectedNbOfTravelSolutions);

```

```

00209
00210 }
00211
00215 BOOST_AUTO_TEST_CASE (airtsp_default_bom_tree_simple_build) {
00216
00217     // State whether the BOM tree should be built-in or parsed from input files
00218     const bool isBuiltin = true;
00219     const bool isWithOnD = false;
00220
00221     // Try to build a travel solution list
00222     unsigned int lNbOfTravelSolutions = 0;
00223     BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
00224         testScheduleHelper (1, " ", " ", isBuiltin, isWithOnD));
00225
00226     // Check the size of the travel solution list
00227     const unsigned int lExpectedNbOfTravelSolutions = 1;
00228     BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
00229         "The number of travel solutions is "
00230         << lNbOfTravelSolutions << ", but it should be equal to "
00231         << lExpectedNbOfTravelSolutions);
00232
00233 }
00234
00238 BOOST_AUTO_TEST_CASE (airtsp_OnD_input_file) {
00239
00240     // Input file names
00241     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00242         "/rds01/schedule05.csv");
00243     const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00244         "/ond01.csv");
00245
00246     // State whether the BOM tree should be built-in or parsed from input files
00247     const bool isBuiltin = false;
00248     const bool isWithOnD = true;
00249
00250     // Try to build a travel solution list
00251     unsigned int lNbOfTravelSolutions = 0;
00252     BOOST_CHECK_NO_THROW (lNbOfTravelSolutions =
00253         testScheduleHelper (2, lScheduleInputFilename,
00254             lODInputFilename,
00255             isBuiltin, isWithOnD));
00256
00257     // Check the size of the travel solution list
00258     const unsigned int lExpectedNbOfTravelSolutions = 1;
00259     BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
00260         "The number of travel solutions is "
00261         << lNbOfTravelSolutions << ", but it should be equal to "
00262         << lExpectedNbOfTravelSolutions);
00263 }
00264
00268 BOOST_AUTO_TEST_CASE (airtsp_missing_OnD_input_file) {
00269
00270     // Input file names
00271     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00272         "/schedule03.csv");
00273     const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00274         "/missingFiles.csv");
00275
00276     // State whether the BOM tree should be built-in or parsed from input files
00277     const bool isBuiltin = false;
00278     const bool isWithOnD = true;
00279
00280     // Try to build a travel solution list
00281     BOOST_CHECK_THROW (testScheduleHelper (3, lScheduleInputFilename,
00282         lODInputFilename,
00283         isBuiltin, isWithOnD),
00284         AIRTSP::OnDInputFileNotFoundException);

```

```
00285 }
00286
00290 BOOST_AUTO_TEST_CASE (airtsp_missing_schedule_input_file) {
00291
00292     // Input file name
00293     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00294                                                         "/missingFiles.csv");
00295
00296     // State whether the BOM tree should be built-in or parsed from input files
00297     const bool isBuiltin = false;
00298     const bool isWithOnD = false;
00299
00300     // Try to build a travel solution list
00301     BOOST_CHECK_THROW (testScheduleHelper (4, lScheduleInputFilename, " ",
00302                                             isBuiltin, isWithOnD),
00303                         AIRTSP::ScheduleInputFileNotFoundException);
00304
00305 }
00306
00310 BOOST_AUTO_TEST_CASE (airtsp_segment_date_not_found) {
00311
00312     // Input file name
00313     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00314                                                         "/scheduleError03.csv");
00315
00316     // State whether the BOM tree should be built-in or parsed from input files
00317     const bool isBuiltin = false;
00318     const bool isWithOnD = false;
00319
00320     // Try to build a travel solution list
00321     BOOST_CHECK_THROW (testScheduleHelper (5, lScheduleInputFilename,
00322                                             " ",
00323                                             isBuiltin, isWithOnD),
00324                         AIRTSP::SegmentDateNotFoundException);
00325
00326
00327 }
00328
00329
00330 // End the test suite
00331 BOOST_AUTO_TEST_SUITE_END()
00332
00333
```

## 26.163 test/airtsp/AirlineScheduleTestSuite.hpp File Reference

```
#include <sstream>
```

```
#include <cppunit/extensions/HelperMacros.h>
```

### Classes

- class [AirlineScheduleTestSuite](#)

### Functions

- [CPPUNIT\\_TEST\\_SUITE\\_REGISTRATION](#) ([AirlineScheduleTestSuite](#))

#### 26.163.1 Function Documentation

##### 26.163.1.1 CPPUNIT\_TEST\_SUITE\_REGISTRATION ([AirlineScheduleTestSuite](#))

**26.164 AirlineScheduleTestSuite.hpp**

```
00001 // STL
00002 #include <sstream>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00006 class AirlineScheduleTestSuite : public CppUnit::TestFixture {
00007     CPPUNIT_TEST_SUITE (AirlineScheduleTestSuite);
00008     // CPPUNIT_TEST (externalMemoryManagement);
00009     CPPUNIT_TEST (scheduleParsing);
00010     CPPUNIT_TEST_SUITE_END ();
00011 public:
00012
00013     void externalMemoryManagement ();
00014     void scheduleParsing ();
00015
00016     AirlineScheduleTestSuite ();
00017
00018 protected:
00019     std::stringstream _describeKey;
00020 };
00021
00022 CPPUNIT_TEST_SUITE_REGISTRATION (AirlineScheduleTestSuite);
```