

AirSched

0.1.2

Generated by Doxygen 1.7.4

Mon Nov 14 2011 13:31:57

Contents

1	AirSched Documentation	1
1.1	Getting Started	1
1.2	AirSched at SourceForge	1
1.3	AirSched Development	1
1.4	External Libraries	2
1.5	Support AirSched	2
1.6	About AirSched	2
2	Configuration helper for AirSched programs	2
3	People	3
3.1	Project Admins	3
3.2	Developers	3
3.3	Retired Developers	4
3.4	Contributors	4
3.5	Distribution Maintainers	4
4	Coding Rules	4
4.1	Default Naming Rules for Variables	4
4.2	Default Naming Rules for Functions	5
4.3	Default Naming Rules for Classes and Structures	5
4.4	Default Naming Rules for Files	5
4.5	Default Functionality of Classes	5
5	Copyright and License	5
5.1	GNU LESSER GENERAL PUBLIC LICENSE	5
5.1.1	Version 2.1, February 1999	6
5.2	Preamble	6
5.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MOD- IFICATION	7
5.3.1	NO WARRANTY	12
5.3.2	END OF TERMS AND CONDITIONS	13
5.4	How to Apply These Terms to Your New Programs	13
6	Documentation Rules	14

6.1	General Rules	14
6.2	File Header	15
6.3	Grouping Various Parts	15
7	Main features	16
7.1	Network generation	16
7.2	Finding travel solutions	16
7.3	Other features	16
8	Make a Difference	16
9	Make a new release	17
9.1	Introduction	17
9.2	Initialisation	17
9.3	Branch creation	17
9.4	Commit and publish the release branch	17
9.5	Update the change-log in the trunk as well	18
9.6	Create distribution packages	18
9.7	Generation the RPM packages	18
9.8	Update distributed change log	19
9.9	Create the binary package, including the documentation	19
9.10	Upload the files to SourceForge	19
9.11	Upload the documentation to SourceForge	19
9.12	Make a new post	20
9.13	Send an email on the announcement mailing-list	20
10	Installation	20
10.1	Table of Contents	20
10.2	Fedora/RedHat Linux distributions	20
10.3	AirSched Requirements	21
10.4	Basic Installation	21
10.5	Compilers and Options	22
10.6	Compiling For Multiple Architectures	23
10.7	Installation Names	23
10.8	Optional Features	24

10.9 Particular systems	25
10.10 Specifying the System Type	26
10.11 Sharing Defaults	26
10.12 Defining Variables	26
10.13 'cmake' Invocation	27
11 Linking with AirSched	31
11.1 Table of Contents	31
11.2 Introduction	31
11.3 Dependencies	31
11.3.1 StdAir	32
11.4 Using the pkg-config command	32
11.5 Using the airsched-config script	33
11.6 M4 macro for the GNU Autotools	33
11.7 Using AirSched with dynamic linking	33
12 Test Rules	34
12.1 The Test File	34
12.2 The Reference File	34
12.3 Testing IT++ Library	34
13 Users Guide	34
13.1 Table of Contents	34
13.2 Introduction	35
13.3 Get Started	35
13.3.1 Get the AirSched library	35
13.3.2 Build the AirSched project	35
13.3.3 Build and Run the Tests	36
13.3.4 Install the AirSched Project (Binaries, Documentation)	36
13.4 Input file of AirSched Project	37
13.5 The schedule BOM Tree	38
13.5.1 Build of the schedule BOM tree	39
13.5.2 Display of the schedule BOM tree	39
13.6 Exploring the Predefined BOM Tree	100
13.6.1 Airline Network BOM Tree	100

13.6.2 Airline Schedule BOM Tree	100
13.7 Extending the BOM Tree	101
13.8 The travel solution calculation procedure	101
14 Supported Systems	101
14.1 Table of Contents	101
14.2 Introduction	101
14.3 AirSched 0.2.x	102
14.3.1 Linux Systems	102
14.3.2 Windows Systems	106
14.3.3 Unix Systems	109
15 AirSched Supported Systems (Previous Releases)	110
15.1 AirSched 3.9.1	110
15.2 AirSched 3.9.0	110
15.3 AirSched 3.8.1	110
16 Tutorials	110
16.1 Table of Contents	110
16.2 Preparing the AirSched Project for Development	110
16.3 Your first networkBuilde	110
16.3.1 Summary of the different steps	110
16.3.2 Result of the Batch Program	111
16.4 Network building with an input file	111
16.4.1 How to build a network input file?	111
16.4.2 Building the BOM tree with an input file	114
16.4.3 Result of the Batch Program	114
17 Command-Line Test to Demonstrate How To Test the AirSched Project	114
18 Directory Hierarchy	117
18.1 Directories	117
19 Namespace Index	117
19.1 Namespace List	117
20 Class Index	118

20.1 Class Hierarchy	118
21 Class Index	122
21.1 Class List	122
22 File Index	126
22.1 File List	126
23 Directory Documentation	129
23.1 test/airsched/ Directory Reference	129
23.2 airsched/ Directory Reference	129
23.3 airsched/basic/ Directory Reference	129
23.4 airsched/batches/ Directory Reference	130
23.5 airsched/bom/ Directory Reference	130
23.6 airsched/command/ Directory Reference	131
23.7 airsched/config/ Directory Reference	131
23.8 airsched/factory/ Directory Reference	131
23.9 airsched/service/ Directory Reference	132
23.10test/ Directory Reference	132
24 Namespace Documentation	132
24.1 airsched Namespace Reference	132
24.1.1 Typedef Documentation	133
24.1.2 Function Documentation	133
24.1.3 Variable Documentation	134
24.2 AIRSCHED Namespace Reference	135
24.2.1 Typedef Documentation	137
24.2.2 Function Documentation	141
24.2.3 Variable Documentation	141
24.3 AIRSCHED::OnDParserHelper Namespace Reference	141
24.3.1 Function Documentation	142
24.3.2 Variable Documentation	144
24.4 AIRSCHED::ScheduleParserHelper Namespace Reference	144
24.4.1 Function Documentation	145
24.4.2 Variable Documentation	147

24.5	boost Namespace Reference	148
24.5.1	Detailed Description	148
24.6	boost::serialization Namespace Reference	148
24.7	stdair Namespace Reference	148
24.7.1	Detailed Description	148
25	Class Documentation	148
25.1	airsched::Airline_T Struct Reference	148
25.1.1	Detailed Description	149
25.1.2	Constructor & Destructor Documentation	149
25.1.3	Member Function Documentation	149
25.1.4	Member Data Documentation	149
25.2	AirlineScheduleTestSuite Class Reference	149
25.2.1	Detailed Description	150
25.2.2	Constructor & Destructor Documentation	150
25.2.3	Member Function Documentation	150
25.2.4	Member Data Documentation	150
25.3	AIRSCHED::AIRSCHED_Service Class Reference	151
25.3.1	Detailed Description	151
25.3.2	Constructor & Destructor Documentation	151
25.3.3	Member Function Documentation	152
25.4	AIRSCHED::AIRSCHED_ServiceContext Class Reference	155
25.4.1	Detailed Description	155
25.4.2	Friends And Related Function Documentation	155
25.5	BomAbstract Class Reference	156
25.6	AIRSCHED::BomDisplay Class Reference	156
25.6.1	Detailed Description	156
25.6.2	Member Function Documentation	156
25.7	CmdAbstract Class Reference	157
25.8	airsched::Date_T Struct Reference	158
25.8.1	Detailed Description	159
25.8.2	Constructor & Destructor Documentation	159
25.8.3	Member Function Documentation	159
25.8.4	Member Data Documentation	159

25.9 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference	160
25.9.1 Detailed Description	161
25.9.2 Constructor & Destructor Documentation	161
25.9.3 Member Function Documentation	161
25.9.4 Member Data Documentation	161
25.10airsched::SearchStringParser::definition< ScannerT > Struct Template Reference	164
25.10.1 Detailed Description	165
25.10.2 Constructor & Destructor Documentation	165
25.10.3 Member Function Documentation	165
25.10.4 Member Data Documentation	165
25.11AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT > Struct Template Reference	167
25.11.1 Detailed Description	168
25.11.2 Constructor & Destructor Documentation	168
25.11.3 Member Function Documentation	168
25.11.4 Member Data Documentation	168
25.12AIRSCHED::ScheduleParserHelper::doEndFlight Struct Reference	169
25.12.1 Detailed Description	170
25.12.2 Constructor & Destructor Documentation	170
25.12.3 Member Function Documentation	170
25.12.4 Member Data Documentation	170
25.13AIRSCHED::OnDParserHelper::doEndOnD Struct Reference	171
25.13.1 Detailed Description	171
25.13.2 Constructor & Destructor Documentation	172
25.13.3 Member Function Documentation	172
25.13.4 Member Data Documentation	172
25.14AIRSCHED::FacAIRSCHEDServiceContext Class Reference	173
25.14.1 Detailed Description	173
25.14.2 Constructor & Destructor Documentation	173
25.14.3 Member Function Documentation	174
25.15AIRSCHED::FacServiceAbstract Class Reference	174
25.15.1 Detailed Description	175
25.15.2 Member Typedef Documentation	175

25.15.3 Constructor & Destructor Documentation	175
25.15.4 Member Function Documentation	176
25.15.5 Member Data Documentation	176
25.16FacServiceAbstract Class Reference	176
25.17AIRSCHED::FareFamilyStruct Struct Reference	176
25.17.1 Detailed Description	177
25.17.2 Constructor & Destructor Documentation	177
25.17.3 Member Function Documentation	177
25.17.4 Member Data Documentation	177
25.18FileNotFoundException Class Reference	178
25.19AIRSCHED::FlagSaver Struct Reference	178
25.19.1 Detailed Description	178
25.19.2 Constructor & Destructor Documentation	178
25.20AIRSCHED::FlightPeriodFileParser Class Reference	179
25.20.1 Detailed Description	179
25.20.2 Constructor & Destructor Documentation	179
25.20.3 Member Function Documentation	180
25.21AIRSCHED::ScheduleParserHelper::FlightPeriodParser Struct Reference	180
25.21.1 Detailed Description	180
25.21.2 Constructor & Destructor Documentation	181
25.21.3 Member Data Documentation	181
25.22AIRSCHED::FlightPeriodStruct Struct Reference	181
25.22.1 Detailed Description	183
25.22.2 Constructor & Destructor Documentation	183
25.22.3 Member Function Documentation	183
25.22.4 Member Data Documentation	185
25.23grammar Class Reference	188
25.24AIRSCHED::InventoryGenerator Class Reference	188
25.24.1 Detailed Description	189
25.24.2 Friends And Related Function Documentation	189
25.25KeyAbstract Class Reference	189
25.26AIRSCHED::LegCabinStruct Struct Reference	189
25.26.1 Detailed Description	190
25.26.2 Member Function Documentation	190

25.26.3 Member Data Documentation	190
25.27AIRSCHED::LegStruct Struct Reference	191
25.27.1 Detailed Description	191
25.27.2 Constructor & Destructor Documentation	192
25.27.3 Member Function Documentation	192
25.27.4 Member Data Documentation	192
25.28AIRSCHED::OnDInputFileNotFoundException Class Reference	193
25.28.1 Detailed Description	194
25.28.2 Constructor & Destructor Documentation	194
25.29AIRSCHED::OnDParser Class Reference	194
25.29.1 Detailed Description	194
25.29.2 Member Function Documentation	194
25.30AIRSCHED::OnDParserHelper::OnDParser Struct Reference	195
25.30.1 Detailed Description	195
25.30.2 Constructor & Destructor Documentation	196
25.30.3 Member Data Documentation	196
25.31AIRSCHED::OnDPeriodFileParser Class Reference	196
25.31.1 Detailed Description	196
25.31.2 Constructor & Destructor Documentation	197
25.31.3 Member Function Documentation	197
25.32AIRSCHED::OnDPeriodGenerator Class Reference	197
25.32.1 Detailed Description	198
25.32.2 Friends And Related Function Documentation	198
25.33AIRSCHED::OnDPeriodStruct Struct Reference	198
25.33.1 Detailed Description	199
25.33.2 Constructor & Destructor Documentation	199
25.33.3 Member Function Documentation	199
25.33.4 Member Data Documentation	200
25.34AIRSCHED::OriginDestinationSet Class Reference	202
25.34.1 Detailed Description	204
25.34.2 Member Typedef Documentation	204
25.34.3 Constructor & Destructor Documentation	204
25.34.4 Member Function Documentation	204
25.34.5 Friends And Related Function Documentation	206

25.34.6 Member Data Documentation	206
25.35AIRSCHED::OriginDestinationSetKey Struct Reference	207
25.35.1 Detailed Description	207
25.35.2 Constructor & Destructor Documentation	208
25.35.3 Member Function Documentation	208
25.35.4 Friends And Related Function Documentation	209
25.36ParserException Class Reference	209
25.37AIRSCHED::OnDParserHelper::ParserSemanticAction Struct Reference	210
25.37.1 Detailed Description	210
25.37.2 Constructor & Destructor Documentation	210
25.37.3 Member Data Documentation	211
25.38AIRSCHED::ScheduleParserHelper::ParserSemanticAction Struct Reference	211
25.38.1 Detailed Description	212
25.38.2 Constructor & Destructor Documentation	212
25.38.3 Member Data Documentation	213
25.39airsched::Passenger_T Struct Reference	213
25.39.1 Detailed Description	214
25.39.2 Member Enumeration Documentation	214
25.39.3 Constructor & Destructor Documentation	214
25.39.4 Member Function Documentation	214
25.39.5 Member Data Documentation	214
25.40airsched::Place_T Struct Reference	215
25.40.1 Detailed Description	215
25.40.2 Constructor & Destructor Documentation	215
25.40.3 Member Function Documentation	216
25.40.4 Member Data Documentation	216
25.41AIRSCHED::ReachableUniverse Class Reference	216
25.41.1 Detailed Description	217
25.41.2 Member Typedef Documentation	217
25.41.3 Constructor & Destructor Documentation	218
25.41.4 Member Function Documentation	218
25.41.5 Friends And Related Function Documentation	220
25.41.6 Member Data Documentation	220

25.42AIRSCHED::ReachableUniverseKey Struct Reference	221
25.42.1 Detailed Description	222
25.42.2 Constructor & Destructor Documentation	222
25.42.3 Member Function Documentation	222
25.42.4 Friends And Related Function Documentation	223
25.43AIRSCHED::ScheduleInputFileNotFoundException Class Reference	223
25.43.1 Detailed Description	224
25.43.2 Constructor & Destructor Documentation	224
25.44AIRSCHED::ScheduleParser Class Reference	224
25.44.1 Detailed Description	225
25.44.2 Member Function Documentation	225
25.45airsched::SearchString_T Struct Reference	225
25.45.1 Detailed Description	226
25.45.2 Constructor & Destructor Documentation	226
25.45.3 Member Function Documentation	226
25.45.4 Member Data Documentation	226
25.46airsched::SearchStringParser Struct Reference	227
25.46.1 Detailed Description	228
25.46.2 Constructor & Destructor Documentation	228
25.46.3 Member Data Documentation	228
25.47AIRSCHED::SegmentCabinStruct Struct Reference	228
25.47.1 Detailed Description	229
25.47.2 Member Function Documentation	229
25.47.3 Member Data Documentation	229
25.48AIRSCHED::SegmentDateNotFoundException Class Reference	230
25.48.1 Detailed Description	230
25.48.2 Constructor & Destructor Documentation	230
25.49AIRSCHED::SegmentPathGenerator Class Reference	231
25.49.1 Detailed Description	231
25.49.2 Member Function Documentation	231
25.50AIRSCHED::SegmentPathPeriod Class Reference	231
25.50.1 Detailed Description	233
25.50.2 Member Typedef Documentation	233
25.50.3 Constructor & Destructor Documentation	233

25.50.4 Member Function Documentation	233
25.50.5 Friends And Related Function Documentation	237
25.50.6 Member Data Documentation	238
25.51AIRSCHED::SegmentPathPeriodKey Struct Reference	238
25.51.1 Detailed Description	239
25.51.2 Constructor & Destructor Documentation	240
25.51.3 Member Function Documentation	240
25.51.4 Friends And Related Function Documentation	243
25.52AIRSCHED::SegmentPathProvider Class Reference	243
25.52.1 Detailed Description	243
25.52.2 Friends And Related Function Documentation	243
25.53AIRSCHED::SegmentPeriodHelper Class Reference	244
25.53.1 Detailed Description	244
25.53.2 Member Function Documentation	244
25.54AIRSCHED::SegmentStruct Struct Reference	245
25.54.1 Detailed Description	245
25.54.2 Member Function Documentation	245
25.54.3 Member Data Documentation	246
25.55ServiceAbstract Class Reference	247
25.56AIRSCHED::ServiceAbstract Class Reference	247
25.56.1 Detailed Description	247
25.56.2 Constructor & Destructor Documentation	247
25.56.3 Member Function Documentation	248
25.57AIRSCHED::Simulator Class Reference	248
25.57.1 Detailed Description	249
25.57.2 Member Function Documentation	249
25.58airsched::store_adult_passenger_type Struct Reference	249
25.58.1 Detailed Description	249
25.58.2 Constructor & Destructor Documentation	250
25.58.3 Member Function Documentation	250
25.58.4 Member Data Documentation	250
25.59airsched::store_airline_code Struct Reference	250
25.59.1 Detailed Description	250
25.59.2 Constructor & Destructor Documentation	251

25.59.3 Member Function Documentation	251
25.59.4 Member Data Documentation	251
25.60airsched::store_airline_name Struct Reference	251
25.60.1 Detailed Description	251
25.60.2 Constructor & Destructor Documentation	252
25.60.3 Member Function Documentation	252
25.60.4 Member Data Documentation	252
25.61airsched::store_airline_sign Struct Reference	252
25.61.1 Detailed Description	252
25.61.2 Constructor & Destructor Documentation	253
25.61.3 Member Function Documentation	253
25.61.4 Member Data Documentation	253
25.62airsched::store_child_passenger_type Struct Reference	253
25.62.1 Detailed Description	253
25.62.2 Constructor & Destructor Documentation	254
25.62.3 Member Function Documentation	254
25.62.4 Member Data Documentation	254
25.63airsched::store_date Struct Reference	254
25.63.1 Detailed Description	254
25.63.2 Constructor & Destructor Documentation	255
25.63.3 Member Function Documentation	255
25.63.4 Member Data Documentation	255
25.64airsched::store_passenger_number Struct Reference	255
25.64.1 Detailed Description	255
25.64.2 Constructor & Destructor Documentation	256
25.64.3 Member Function Documentation	256
25.64.4 Member Data Documentation	256
25.65airsched::store_pet_passenger_type Struct Reference	256
25.65.1 Detailed Description	256
25.65.2 Constructor & Destructor Documentation	257
25.65.3 Member Function Documentation	257
25.65.4 Member Data Documentation	257
25.66airsched::store_place_element Struct Reference	257
25.66.1 Detailed Description	257

25.66.2 Constructor & Destructor Documentation	258
25.66.3 Member Function Documentation	258
25.66.4 Member Data Documentation	258
25.67AIRSCHED::OnDParserHelper::storeAirlineCode Struct Reference	258
25.67.1 Detailed Description	259
25.67.2 Constructor & Destructor Documentation	259
25.67.3 Member Function Documentation	259
25.67.4 Member Data Documentation	259
25.68AIRSCHED::ScheduleParserHelper::storeAirlineCode Struct Reference	260
25.68.1 Detailed Description	260
25.68.2 Constructor & Destructor Documentation	260
25.68.3 Member Function Documentation	260
25.68.4 Member Data Documentation	261
25.69AIRSCHED::ScheduleParserHelper::storeBoardingTime Struct Reference	261
25.69.1 Detailed Description	262
25.69.2 Constructor & Destructor Documentation	262
25.69.3 Member Function Documentation	262
25.69.4 Member Data Documentation	262
25.70AIRSCHED::ScheduleParserHelper::storeCapacity Struct Reference	263
25.70.1 Detailed Description	263
25.70.2 Constructor & Destructor Documentation	263
25.70.3 Member Function Documentation	263
25.70.4 Member Data Documentation	264
25.71AIRSCHED::OnDParserHelper::storeClassCode Struct Reference	264
25.71.1 Detailed Description	265
25.71.2 Constructor & Destructor Documentation	265
25.71.3 Member Function Documentation	265
25.71.4 Member Data Documentation	265
25.72AIRSCHED::ScheduleParserHelper::storeClasses Struct Reference	265
25.72.1 Detailed Description	266
25.72.2 Constructor & Destructor Documentation	266
25.72.3 Member Function Documentation	266
25.72.4 Member Data Documentation	266

25.73AIRSCHED::ScheduleParserHelper::storeDateRangeEnd Struct Reference	267
25.73.1 Detailed Description	267
25.73.2 Constructor & Destructor Documentation	268
25.73.3 Member Function Documentation	268
25.73.4 Member Data Documentation	268
25.74AIRSCHED::OnDParserHelper::storeDateRangeEnd Struct Reference	268
25.74.1 Detailed Description	269
25.74.2 Constructor & Destructor Documentation	269
25.74.3 Member Function Documentation	269
25.74.4 Member Data Documentation	270
25.75AIRSCHED::ScheduleParserHelper::storeDateRangeStart Struct Reference	270
25.75.1 Detailed Description	270
25.75.2 Constructor & Destructor Documentation	271
25.75.3 Member Function Documentation	271
25.75.4 Member Data Documentation	271
25.76AIRSCHED::OnDParserHelper::storeDateRangeStart Struct Reference	271
25.76.1 Detailed Description	272
25.76.2 Constructor & Destructor Documentation	272
25.76.3 Member Function Documentation	272
25.76.4 Member Data Documentation	272
25.77AIRSCHED::OnDParserHelper::storeDestination Struct Reference	273
25.77.1 Detailed Description	273
25.77.2 Constructor & Destructor Documentation	273
25.77.3 Member Function Documentation	274
25.77.4 Member Data Documentation	274
25.78AIRSCHED::ScheduleParserHelper::storeDow Struct Reference	274
25.78.1 Detailed Description	275
25.78.2 Constructor & Destructor Documentation	275
25.78.3 Member Function Documentation	275
25.78.4 Member Data Documentation	275
25.79AIRSCHED::ScheduleParserHelper::storeElapsedTime Struct Reference	276
25.79.1 Detailed Description	276

25.79.2 Constructor & Destructor Documentation	276
25.79.3 Member Function Documentation	277
25.79.4 Member Data Documentation	277
25.80AIRSCHED::OnDParserHelper::storeEndRangeTime Struct Reference	277
25.80.1 Detailed Description	278
25.80.2 Constructor & Destructor Documentation	278
25.80.3 Member Function Documentation	278
25.80.4 Member Data Documentation	278
25.81AIRSCHED::ScheduleParserHelper::storeFamilyCode Struct Reference	279
25.81.1 Detailed Description	279
25.81.2 Constructor & Destructor Documentation	279
25.81.3 Member Function Documentation	279
25.81.4 Member Data Documentation	280
25.82AIRSCHED::ScheduleParserHelper::storeFCClasses Struct Reference	280
25.82.1 Detailed Description	281
25.82.2 Constructor & Destructor Documentation	281
25.82.3 Member Function Documentation	281
25.82.4 Member Data Documentation	281
25.83AIRSCHED::ScheduleParserHelper::storeFlightNumber Struct Reference	282
25.83.1 Detailed Description	282
25.83.2 Constructor & Destructor Documentation	282
25.83.3 Member Function Documentation	283
25.83.4 Member Data Documentation	283
25.84AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint Struct Reference	283
25.84.1 Detailed Description	284
25.84.2 Constructor & Destructor Documentation	284
25.84.3 Member Function Documentation	284
25.84.4 Member Data Documentation	284
25.85AIRSCHED::ScheduleParserHelper::storeLegCabinCode Struct Reference	285
25.85.1 Detailed Description	285
25.85.2 Constructor & Destructor Documentation	285
25.85.3 Member Function Documentation	286

25.85.4 Member Data Documentation	286
25.86AIRSCHED::ScheduleParserHelper::storeLegOffPoint Struct Reference	286
25.86.1 Detailed Description	287
25.86.2 Constructor & Destructor Documentation	287
25.86.3 Member Function Documentation	287
25.86.4 Member Data Documentation	287
25.87AIRSCHED::ScheduleParserHelper::storeOffTime Struct Reference	288
25.87.1 Detailed Description	288
25.87.2 Constructor & Destructor Documentation	288
25.87.3 Member Function Documentation	289
25.87.4 Member Data Documentation	289
25.88AIRSCHED::OnDParserHelper::storeOrigin Struct Reference	289
25.88.1 Detailed Description	290
25.88.2 Constructor & Destructor Documentation	290
25.88.3 Member Function Documentation	290
25.88.4 Member Data Documentation	290
25.89AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference	291
25.89.1 Detailed Description	291
25.89.2 Constructor & Destructor Documentation	291
25.89.3 Member Function Documentation	292
25.89.4 Member Data Documentation	292
25.90AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode Struct Reference	292
25.90.1 Detailed Description	293
25.90.2 Constructor & Destructor Documentation	293
25.90.3 Member Function Documentation	293
25.90.4 Member Data Documentation	293
25.91AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint Struct Reference	294
25.91.1 Detailed Description	294
25.91.2 Constructor & Destructor Documentation	294
25.91.3 Member Function Documentation	295
25.91.4 Member Data Documentation	295

25.92AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity Struct Reference	295
25.92.1 Detailed Description	296
25.92.2 Constructor & Destructor Documentation	296
25.92.3 Member Function Documentation	296
25.92.4 Member Data Documentation	296
25.93AIRSCHED::OnDParserHelper::storeStartRangeTime Struct Reference	297
25.93.1 Detailed Description	297
25.93.2 Constructor & Destructor Documentation	297
25.93.3 Member Function Documentation	298
25.93.4 Member Data Documentation	298
25.94StructAbstract Class Reference	298
25.95TestFixture Class Reference	298
25.96AIRSCHED::TravelSolutionParser Class Reference	299
25.96.1 Detailed Description	299
25.96.2 Member Function Documentation	299
26 File Documentation	300
26.1 airsched/AIRSCHED_Service.hpp File Reference	300
26.2 AIRSCHED_Service.hpp	300
26.3 airsched/AIRSCHED_Types.hpp File Reference	302
26.4 AIRSCHED_Types.hpp	302
26.5 airsched/basic/BasConst.cpp File Reference	303
26.6 BasConst.cpp	303
26.7 airsched/basic/BasConst_AIRSCHED_Service.hpp File Reference	304
26.8 BasConst_AIRSCHED_Service.hpp	304
26.9 airsched/basic/BasConst_General.hpp File Reference	304
26.10BasConst_General.hpp	304
26.11airsched/basic/BasParserTypes.hpp File Reference	305
26.12BasParserTypes.hpp	306
26.13airsched/batches/airsched.cpp File Reference	307
26.13.1 Typedef Documentation	308
26.13.2 Function Documentation	308
26.13.3 Variable Documentation	309

26.14airsched.cpp	310
26.15airsched/batches/BookingRequestParser.cpp File Reference	316
26.15.1 Define Documentation	318
26.15.2 Typedef Documentation	318
26.16BookingRequestParser.cpp	318
26.17airsched/batches/BookingRequestParser.hpp File Reference	324
26.18BookingRequestParser.hpp	324
26.19airsched/bom/AirportList.hpp File Reference	327
26.20AirportList.hpp	327
26.21airsched/bom/BomDisplay.cpp File Reference	327
26.22BomDisplay.cpp	328
26.23airsched/bom/BomDisplay.hpp File Reference	329
26.24BomDisplay.hpp	330
26.25airsched/bom/FareFamilyStruct.cpp File Reference	330
26.26FareFamilyStruct.cpp	330
26.27airsched/bom/FareFamilyStruct.hpp File Reference	331
26.28FareFamilyStruct.hpp	331
26.29airsched/bom/FlightPeriodStruct.cpp File Reference	332
26.30FlightPeriodStruct.cpp	332
26.31airsched/bom/FlightPeriodStruct.hpp File Reference	336
26.32FlightPeriodStruct.hpp	337
26.33airsched/bom/LegCabinStruct.cpp File Reference	338
26.34LegCabinStruct.cpp	339
26.35airsched/bom/LegCabinStruct.hpp File Reference	339
26.36LegCabinStruct.hpp	340
26.37airsched/bom/LegStruct.cpp File Reference	340
26.38LegStruct.cpp	340
26.39airsched/bom/LegStruct.hpp File Reference	342
26.40LegStruct.hpp	342
26.41airsched/bom/OnDPeriodStruct.cpp File Reference	343
26.42OnDPeriodStruct.cpp	343
26.43airsched/bom/OnDPeriodStruct.hpp File Reference	345
26.44OnDPeriodStruct.hpp	345
26.45airsched/bom/OriginDestinationSet.cpp File Reference	346

26.46OriginDestinationSet.cpp	346
26.47airsched/bom/OriginDestinationSet.hpp File Reference	348
26.48OriginDestinationSet.hpp	348
26.49airsched/bom/OriginDestinationSetKey.cpp File Reference	350
26.50OriginDestinationSetKey.cpp	351
26.51airsched/bom/OriginDestinationSetKey.hpp File Reference	352
26.52OriginDestinationSetKey.hpp	352
26.53airsched/bom/OriginDestinationSetTypes.hpp File Reference	354
26.54OriginDestinationSetTypes.hpp	354
26.55airsched/bom/ReachableUniverse.cpp File Reference	355
26.56ReachableUniverse.cpp	355
26.57airsched/bom/ReachableUniverse.hpp File Reference	356
26.58ReachableUniverse.hpp	357
26.59airsched/bom/ReachableUniverseKey.cpp File Reference	359
26.60ReachableUniverseKey.cpp	359
26.61airsched/bom/ReachableUniverseKey.hpp File Reference	361
26.62ReachableUniverseKey.hpp	361
26.63airsched/bom/ReachableUniverseTypes.hpp File Reference	363
26.64ReachableUniverseTypes.hpp	363
26.65airsched/bom/SegmentCabinStruct.cpp File Reference	363
26.66SegmentCabinStruct.cpp	364
26.67airsched/bom/SegmentCabinStruct.hpp File Reference	364
26.68SegmentCabinStruct.hpp	365
26.69airsched/bom/SegmentPathPeriod.cpp File Reference	366
26.70SegmentPathPeriod.cpp	366
26.71airsched/bom/SegmentPathPeriod.hpp File Reference	371
26.72SegmentPathPeriod.hpp	372
26.73airsched/bom/SegmentPathPeriodKey.cpp File Reference	374
26.74SegmentPathPeriodKey.cpp	375
26.75airsched/bom/SegmentPathPeriodKey.hpp File Reference	376
26.76SegmentPathPeriodKey.hpp	377
26.77airsched/bom/SegmentPathPeriodTypes.hpp File Reference	379
26.78SegmentPathPeriodTypes.hpp	380
26.79airsched/bom/SegmentPeriodHelper.cpp File Reference	380

26.80SegmentPeriodHelper.cpp	381
26.81airsched/bom/SegmentPeriodHelper.hpp File Reference	382
26.82SegmentPeriodHelper.hpp	382
26.83airsched/bom/SegmentStruct.cpp File Reference	383
26.84SegmentStruct.cpp	383
26.85airsched/bom/SegmentStruct.hpp File Reference	384
26.86SegmentStruct.hpp	385
26.87airsched/command/InventoryGenerator.cpp File Reference	385
26.88InventoryGenerator.cpp	386
26.89airsched/command/InventoryGenerator.hpp File Reference	388
26.90InventoryGenerator.hpp	388
26.91airsched/command/OnDParser.cpp File Reference	389
26.92OnDParser.cpp	389
26.93airsched/command/OnDParser.hpp File Reference	390
26.94OnDParser.hpp	390
26.95airsched/command/OnDParserHelper.cpp File Reference	391
26.96OnDParserHelper.cpp	392
26.97airsched/command/OnDParserHelper.hpp File Reference	398
26.98OnDParserHelper.hpp	399
26.99airsched/command/OnDPeriodGenerator.cpp File Reference	401
26.100OnDPeriodGenerator.cpp	402
26.101airsched/command/OnDPeriodGenerator.hpp File Reference	402
26.102OnDPeriodGenerator.hpp	403
26.103airsched/command/ScheduleParser.cpp File Reference	403
26.104ScheduleParser.cpp	404
26.105airsched/command/ScheduleParser.hpp File Reference	404
26.106ScheduleParser.hpp	405
26.107airsched/command/ScheduleParserHelper.cpp File Reference	405
26.108ScheduleParserHelper.cpp	406
26.109airsched/command/ScheduleParserHelper.hpp File Reference	418
26.110ScheduleParserHelper.hpp	419
26.111airsched/command/SegmentPathGenerator.cpp File Reference	422
26.112SegmentPathGenerator.cpp	422
26.113airsched/command/SegmentPathGenerator.hpp File Reference	429

26.11	SegmentPathGenerator.hpp	430
26.11	airsched/command/SegmentPathProvider.cpp File Reference	431
26.11	SegmentPathProvider.cpp	431
26.11	airsched/command/SegmentPathProvider.hpp File Reference	434
26.11	SegmentPathProvider.hpp	434
26.11	airsched/command/Simulator.cpp File Reference	435
26.12	Simulator.cpp	435
26.12	airsched/command/Simulator.hpp File Reference	436
26.12	Simulator.hpp	436
26.12	airsched/command/TravelSolutionParser.cpp File Reference	437
26.12	TravelSolutionParser.cpp	437
26.12	airsched/command/TravelSolutionParser.hpp File Reference	440
26.12	TravelSolutionParser.hpp	440
26.12	airsched-paths.hpp	442
26.12	airsched/config/airsched-paths.hpp.in File Reference	442
26.12	airsched-paths.hpp.in	442
26.13	airsched/factory/FacAIRSCHEDServiceContext.cpp File Reference	443
26.13	FacAIRSCHEDServiceContext.cpp	443
26.13	airsched/factory/FacAIRSCHEDServiceContext.hpp File Reference	444
26.13	FacAIRSCHEDServiceContext.hpp	444
26.13	airsched/factory/FacServiceAbstract.cpp File Reference	445
26.13	FacServiceAbstract.cpp	445
26.13	airsched/factory/FacServiceAbstract.hpp File Reference	445
26.13	FacServiceAbstract.hpp	446
26.13	airsched/service/AIRSCHED_Service.cpp File Reference	446
26.13	AIRSCHED_Service.cpp	447
26.14	airsched/service/AIRSCHED_ServiceContext.cpp File Reference	453
26.14	AIRSCHED_ServiceContext.cpp	453
26.14	airsched/service/AIRSCHED_ServiceContext.hpp File Reference	454
26.14	AIRSCHED_ServiceContext.hpp	455
26.14	airsched/service/ServiceAbstract.cpp File Reference	456
26.14	ServiceAbstract.cpp	456
26.14	airsched/service/ServiceAbstract.hpp File Reference	456
26.14	Function Documentation	457

26.143	ServiceAbstract.hpp	457
26.144	doc/local/authors.doc File Reference	458
26.144	doc/local/codingrules.doc File Reference	458
26.150	doc/local/copyright.doc File Reference	458
26.151	doc/local/documentation.doc File Reference	458
26.152	doc/local/features.doc File Reference	459
26.153	doc/local/help_wanted.doc File Reference	459
26.154	doc/local/howto_release.doc File Reference	459
26.155	doc/local/index.doc File Reference	459
26.156	doc/local/installation.doc File Reference	459
26.157	doc/local/linking.doc File Reference	459
26.158	doc/local/test.doc File Reference	459
26.158	doc/local/users_guide.doc File Reference	459
26.160	doc/local/verification.doc File Reference	459
26.161	doc/tutorial/tutorial.doc File Reference	459
26.162	test/airsched/AirlineScheduleTestSuite.cpp File Reference	459
26.163	AirlineScheduleTestSuite.cpp	459
26.164	test/airsched/AirlineScheduleTestSuite.hpp File Reference	461
	26.164.1 Function Documentation	462
26.165	AirlineScheduleTestSuite.hpp	462

1 AirSched Documentation

1.1 Getting Started

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

1.2 AirSched at SourceForge

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

1.3 AirSched Development

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

1.4 External Libraries

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

1.5 Support AirSched

1.6 About AirSched

AirSched 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. AirSched mainly targets simulation purposes. [N](#)

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

AirSched 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 AirSched 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 AirSched.

2 Configuration helper for AirSched programs

```

*/
#ifndef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_PATHS_HPP__

#define PACKAGE "airsched"
#define PACKAGE_NAME "AIRSCHED"
#define PACKAGE_VERSION "0.1.2"
#define PREFIXDIR "/usr"
#define EXEC_PREFIX "/usr"
#define BINDIR "/usr/bin"
#define LIBDIR "/usr/lib"
#define LIBEXECDIR "/usr/libexec"
#define SBINDIR "/usr/sbin"
#define SYSCONFDIR "/usr/etc"
#define INCLUDEDIR "/usr/include"
#define DATAROOTDIR "/usr/share"
#define DATADIR "/usr/share"
#define DOCDIR "/usr/share/doc/airsched-0.1.2"
#define MANDIR "/usr/share/man"
#define INFODIR "/usr/share/info"
#define HTMLDIR "/usr/share/doc/airsched-0.1.2/html"
#define PDFDIR "/usr/share/doc/airsched-0.1.2/html"
#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

#endif // __AIRSCHED_PATHS_HPP__

/*!

*/
#ifndef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_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 // __AIRSCHED_PATHS_HPP__

/*!
```

3 People

3.1 Project Admins

- Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Anh Quan Nguyen <quannaus@users.sourceforge.net> (N)

3.2 Developers

- Anh Quan Nguyen <quannaus@users.sourceforge.net> (N)
- Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Gabrielle Sabatier <gsabatier@users.sourceforge.net> (N)

3.3 Retired Developers

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

3.4 Contributors

- Emmanuel Bastien <ebastien@users.sourceforge.net> (N)
- Christophe Lacombe <ddtof@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 AirSched 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 AirSched 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.
*
* -----
*
* AirSched - C++ Airline Schedule Management 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 AirSched 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 AirSched can do for you. Ask what you can do for AirSched.

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

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

9 Make a new release

9.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of AirSched 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://air-sched.git.sourceforge.net/gitroot/air-sched/air-sched airschedgit
cd airschedgit
git checkout trunk
```


9.3 Branch creation

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

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

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

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

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

```
vi ChangeLog
vi airsched.spec
```

9.4 Commit and publish the release branch

Commit the new release:

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

9.5 Update the change-log in the trunk as well

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

```
cd ~/dev/sim/airschedgit
git checkout trunk
vi ChangeLog
vi airsched.spec
```

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

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

9.6 Create distribution packages

Create the distribution packages using the following command:

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

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

9.7 Generation the RPM packages

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

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

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

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

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 [AirSched'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:

```
make package
```

The output binary package will be named, for instance, `airsched-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 Upload the documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with `rsync` and `SSH`:

```
cd ~/dev/sim/airschedgit
git checkout 0.5.0
rsync -aiv doc/html/ doc/latex/refman.pdf joe,airsched@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`.

9.12 Make a new post

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

9.13 Send an email on the announcement mailing-list

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

10 Installation

10.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [AirSched 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 airsched-devel airsched-doc
```

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

10.3 AirSched Requirements

AirSched 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``)
- [SOI](#) - 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 AirSched.

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/airsched-99.99.99 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE=Debug
-- 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: 6100bb1479e9c72f807a60067138dfelb71c7 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 'airschedlib' to CXX
-- Test 'AirlineScheduleTestSuite' to be built with 'AirlineScheduleTestSuite.cpp'
--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : aairsched
-- PACKAGE_PRETTY_NAME ..... : AirSched
-- PACKAGE ..... : aairsched
-- PACKAGE_NAME ..... : AIRSCHED
-- 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 ..... : aairsched
-- Libraries to build/install ..... : aairschedlib
-- Binaries to build/install ..... : aairsched
-- Modules to test ..... : aairsched
-- Binaries to test ..... : AirlineScheduleTestSuitetst
--
-- * Module ..... : aairsched
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : aairschedlib
--   + Executables to build/install : aairsched
--   + Tests to perform ..... : AirlineScheduleTestSuitetst
--
-- 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/airsched/airschedgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/user/dev/deliveries/airsched-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/airsched-99.99.99/lib64
-- INSTALL_BIN_DIR ..... : /home/user/dev/deliveries/airsched-99.99.99/bin

```

```
-- INSTALL_INCLUDE_DIR ..... : /home/user/dev/deliveries/airsched-99.99.99/include
-- INSTALL_DATA_DIR ..... : /home/user/dev/deliveries/airsched-99.99.99/share
-- INSTALL_SAMPLE_DIR ..... : /home/user/dev/deliveries/airsched-99.99.99/share/airsched/s
-- INSTALL_DOC ..... : ON
--
-- -----
-- --- Packaging Configuration ---
-- -----
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot m
-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/airsched/airschedgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/airsched/airschedgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : airsched-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;files
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/l
--
-- * 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;stdairuiclib
--   - 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/airsched/airschedgithub/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_airsched
[ 96%] Built target airschedlib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airschedtst
Test project /home/dan/dev/sim/airsched/airschedgithub/build/test/airsched
  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_airschedtst
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/airschedgit
rm -rf build && mkdir build
cd build
```

to remove everything.

11 Linking with AirSched

11.1 Table of Contents

- [Introduction](#)
- [Dependencies](#)
- [Using the pkg-config command](#)
- [Using the airsched-config script](#)

- [M4 macro for the GNU Autotools](#)
- [Using AirSched with dynamic linking](#)

11.2 Introduction

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

11.3 Dependencies

The AirSched 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: AirSched 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 AirSched based program 'my_prog.cpp', you should use the following command:

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

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

11.5 Using the airsched-config script

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

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

```
airsched-config --help
```

If the 'airsched-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 AirSched, namely 'airsched.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to the 'AM_PATH_AirSched' macro (specified in the M4 macro file), the following Makefile variables are then defined:

- 'AirSched_VERSION' (e.g., defined to 0.23.0)
- 'AirSched_CFLAGS' (e.g., defined to '-I\${prefix}/include')
- 'AirSched_LIBS' (e.g., defined to '-L\${prefix}/lib -lairsched')

11.7 Using AirSched with dynamic linking

When using static linking some of the library routines in AirSched 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 AirSched library file during your program execution. If you install the AirSched 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=<AirSched 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 AirSched library](#)
 - [Build the AirSched project](#)
 - [Build and Run the Tests](#)
 - [Install the AirSched Project \(Binaries, Documentation\)](#)
- [Input file of AirSched 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 `AirSched` library contains classes for airline business management. This document does not cover all the aspects of the `AirSched` library. It does however explain the most important things you need to know in order to start using `AirSched`.

13.3 Get Started

13.3.1 Get the AirSched library

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

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://air-sched.git.sourceforge.net/gitroot/air-sched/air-sched airschedgit
cd airschedgit
git checkout trunk
```

13.3.2 Build the AirSched project

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

```
cd ~/dev/sim/airschedgit
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=~/dev/deliveries/airsched-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 AirSched project, the following commands run the tests:

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

As a result, you should obtain a similar report:

```
[ 0%] Built target hdr_cfg_airsched
[ 96%] Built target airschedlib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airschedtst
Test project /home/dan/dev/sim/airsched/airschedgithub/build/test/airsched
  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_airschedtst
Scanning dependencies of target check
[100%] Built target check
```

13.3.4 Install the AirSched Project (Binaries, Documentation)

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

```
cd ~/dev/sim/airschedgit
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/airsched-0.5.0
```

To generate the AirSched project documentation, the commands are:

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

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

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

13.4 Input file of AirSched Project

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

```
// Flights:   AirlineCode; FlightNumber; Date-Range; ; DOW; Legs; Segments;
// Legs:      BoardPoint; OffPoint; BoardTime; ArrivalDateOffSet; ArrivalTime;
//            ElapsedTime; LegCabins;
// LegCabins: CabinCode; Capacity;
// 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 [AIRSCHED::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 `AIRSCHED::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 AirSched 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 `AIRSCHED::AIRSCHED_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 `AIRSCHED::AIRSCHED_Service::buildSampleBom()` method:

```
void buildSampleBom();
```

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

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

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::airsched.cpp` program:

When the default BOM tree is used (`-b/--builtin` option of the main program `airsched.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, 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, 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, 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,
*****
*****
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,
*****
*****
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, 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, 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, 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, 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, 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,
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, 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,
SQ11 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,
SQ11 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,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 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,
SQ11 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,
SQ11 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: SQ11, 2010-Feb-11
*****

```



```

*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQ11 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,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
(pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
0, 0, 0, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
0, 0, 0, 0,
*****
*****
FlightDate: SQ11, 2010-Feb-12
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, 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, 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, 0, 300,
      9, 0, 0, 0, 0, 0,
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, 0, 0, 0, 0, 300, 0,
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
FlightDate: SQ11, 2010-Feb-14
*****

```

Leg-Dates:

Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, Elapsed, Distance, Capacity,
 SQ11 2010-Feb-14, SIN-BKK, 2010-Feb-14, 08:20:00, 2010-Feb-14, 11:00:00, 07:40:00, 0, -05:00:00, 6300, 0,

LegCabins:

Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
 SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 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, El
 apsed, 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, Comm
 Space, 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, 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, OfferedCAP, 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, 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, OffedCAP, 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, 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, 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, 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, 0, 200,
      9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-21
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-21, SIN-HND, 2010-Jan-21, 09:20:00, 2010-Jan-21, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
```



```

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

```

```
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-23
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-23, SIN-HND, 2010-Jan-23, 09:20:00, 2010-Jan-23, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Jan-24
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, 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, 0, 200,
    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,
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, 0, 200,
    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,
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, 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, 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, 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, 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,
SQ12 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,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-05
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
SQ12 2010-Feb-05, SIN-HND, 2010-Feb-05, 09:20:00, 2010-Feb-05, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm

```

```

    Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
    9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
    (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
    0, 0, 0, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
    0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-06
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, 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, 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, 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, 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, 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, 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,

```

```
SQL2 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,
SQL2 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,
SQL2 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,
SQL2 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 0,
*****
*****
FlightDate: SQ12, 2010-Feb-15
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-15, SIN-HND, 2010-Feb-15, 09:20:00, 2010-Feb-15, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 2, 0, 0, 0, 0, 200, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
      0, 0, 0, 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, 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, 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, El
apsed, 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, Comm
Space, 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, El
apsed, 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, Comm
Space, 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,
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, 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, 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, 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, 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, OffedCAP, 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, 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, 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

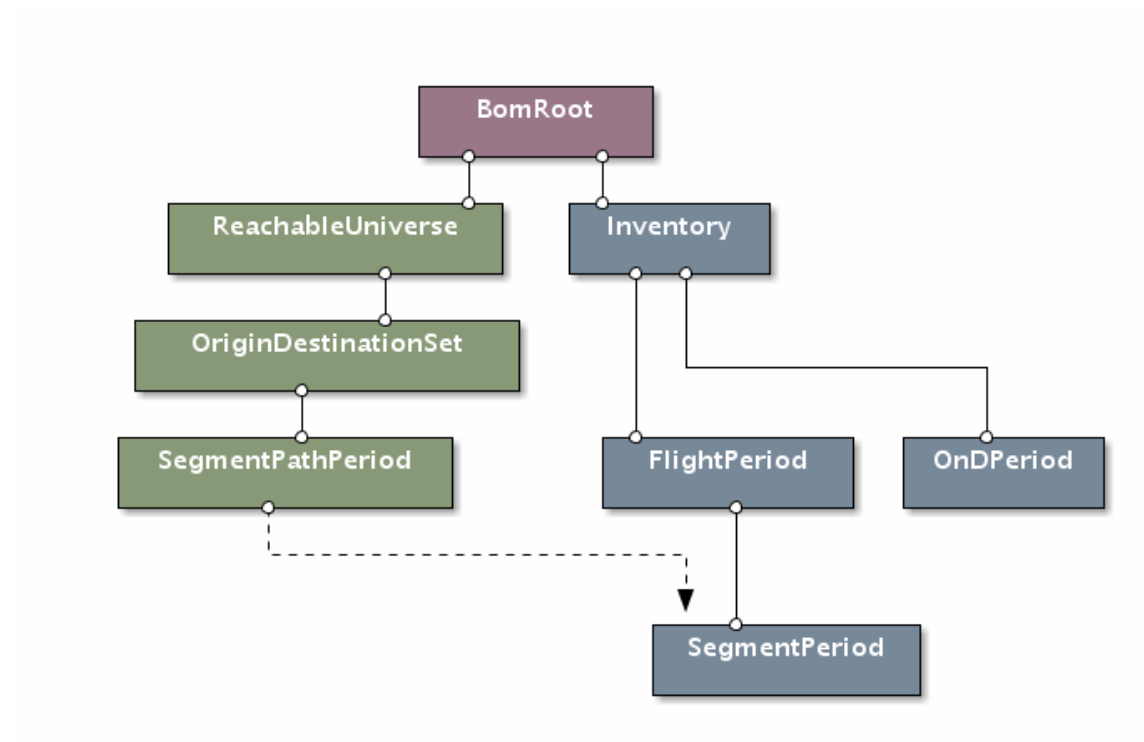


Figure 2: AirSched BOM tree

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

13.6.1 Airline Network BOM Tree

- `AIRSCHEd::ReachableUniverse`
- `AIRSCHEd::OriginDestinationSet`
- `AIRSCHEd::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 AirSched aims at calculating a list of `travel solutions` for every incoming `booking request`.

14 Supported Systems

14.1 Table of Contents

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

14.2 Introduction

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

14.3 AirSched 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
- **AirSched 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:** AirSched 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
- **AirSched 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
```

AirSched 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
- **AirSched 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
```

AirSched 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
- **AirSched 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:** AirSched 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
- **AirSched 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
```

AirSched configured with:

```
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

14.3.1.6 Red Hat Enterprise Linux with AirSched 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)
- **AirSched release:** 0.2.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirSched 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
- **AirSched 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:** AirSched 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
- **AirSched 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:** AirSched 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)
- **AirSched 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. AirSched 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)
- **AirSched release:** 0.2.1
- **External Libraries:** Installed from Cygwin's repository:

```
- fftw-3.0.1-2
- fftw-dev-3.0.1-1
```

ATLAS BLAS and LAPACK libraries from AirSched External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```

- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirSched 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)
- **AirSched 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. AirSched 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)
- **AirSched 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. AirSched 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 AirSched 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)
- **AirSched release:** 0.2.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirSched External 2.2.0 package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. AirSched 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
- **AirSched 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 AirSched based programs compiled and run with success.
- **Comments:** Only static library can be built. AirSched built by opening the "win32\airsched.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 AirSched External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5
- **AirSched release:** 0.2.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from AirSched 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:** AirSched 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 AirSched Supported Systems (Previous Releases)

15.1 AirSched 3.9.1

15.2 AirSched 3.9.0

15.3 AirSched 3.8.1

16 Tutorials

16.1 Table of Contents

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

16.2 Preparing the AirSched Project for Development

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

16.3 Your first networkBuilde

16.3.1 Summary of the different steps

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

First, we instanciate the `AIRSCHEd_Service` object:

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

```
stdair::TravelSolutionList_T lTravelSolutionList;  
airschedService.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 `AirSched.cpp` program is run (with the `-b` option), the log output file should look like:

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

and after the network building:

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

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

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

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 :

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd;
//      DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase;
//      SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments
// 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:

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd;
//      DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase;
//      SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments
// 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:

```

// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart; DateRangeEnd;
//      DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode; Channel; AdvancePurchase;
//      SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; nb Segments
// 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 `AirSched.cpp` program is run with the `-f` option linking with the file built just above:

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

the last lines of the log output should look like:

```

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

```

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

17 Command-Line Test to Demonstrate How To Test the AirSched Project

```

*/
// //////////////////////////////////////
// Import section
// //////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE InventoryTestSuite
#include <boost/test/unit_test.hpp>

```

```

// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
// AirSched
#include <airsched/AIRSCHEM_Service.hpp>
#include <airsched/config/airsched-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() {
    }
};

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

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

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (airsched_simple_inventory_sell) {

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

    // Output log File
    const stdair::Filename_T lLogFilename ("AirlineScheduleTestSuite.log");

    // Check that the file path given as input corresponds to an actual file
    bool doesExistAndIsReadable =
        stdair::BasFileMgr::doesExistAndIsReadable (lScheduleInputFilename);
    BOOST_CHECK_MESSAGE (doesExistAndIsReadable == true,
        "The '" << lScheduleInputFilename
        << "' input file can not be open and read");

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

    // Instantiate the AirSched service

```

```

const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
AIRSCHED::AIRSCHED_Service airschedService (lLogParams);

// Build the BOM tree from parsing input files
airschedService.parseAndLoad (lScheduleInputFilename);

// Create an empty booking request structure
// \todo: fill the booking request structure from the input parameters
const stdair::AirportCode_T lOrigin ("NCE");
const stdair::AirportCode_T lDestination ("BKK");
const stdair::AirportCode_T lPOS ("NCE");
const stdair::Date_T lPreferredDepartureDate(2007, boost::gregorian::Apr, 21);
const stdair::Date_T lRequestDate (2007, boost::gregorian::Mar, 21);
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::BookingRequestStruct lBookingRequest (lOrigin, lDestination,
                                                    lPOS,
                                                    lPreferredDepartureDate,
                                                    lRequestDateTime,
                                                    lPreferredCabin,
                                                    lPartySize, lChannel,
                                                    lTripType, lStayDuration,
                                                    lFrequentFlyerType,
                                                    lPreferredDepartureTime,
                                                    lWTP, lValueOfTime);

//
stdair::TravelSolutionList_T lTravelSolutionList;
airschedService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
const unsigned int lNbOfTravelSolutions = lTravelSolutionList.size();

// \todo: change the expected number of travel solutions to the actual number
const unsigned int lExpectedNbOfTravelSolutions = 4;

// DEBUG
STDAIR_LOG_DEBUG ("Number of travel solutions for the booking request '"
                  << lBookingRequest.describe() << "' : "
                  << lNbOfTravelSolutions << ". It is expected to be "
                  << lExpectedNbOfTravelSolutions << ".");

BOOST_CHECK_EQUAL (lNbOfTravelSolutions, lExpectedNbOfTravelSolutions);

BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
                  "The number of travel solutions for the booking request '"
                  << lBookingRequest.describe() << "' is equal to "
                  << lNbOfTravelSolutions << ", but it should be equal to "
                  << lExpectedNbOfTravelSolutions);

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

```

```
// 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:

airsched	129
basic	129
batches	130
bom	130
command	131
config	131
factory	131
service	132
test	132
airsched	129

19 Namespace Index

19.1 Namespace List

Here is a list of all namespaces with brief descriptions:

airsched	132
AIRSCHEd	135
AIRSCHEd::OnDParserHelper	141
AIRSCHEd::ScheduleParserHelper	144
boost (Forward declarations)	148
boost::serialization	148

[stdair](#) (Forward declarations)

[148](#)

20 Class Index

20.1 Class Hierarchy

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

airsched::Airline_T

[148](#)

AIRSCHEd::AIRSCHEd_Service

[151](#)

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_ostringstream< char >
std::basic_ostringstream< wchar_t >
std::basic_string< char >
std::basic_string< wchar_t >
std::basic_stringstream< char >
std::basic_stringstream< wchar_t >

BomAbstract

[156](#)

AIRSCHEd::OriginDestinationSet

[202](#)

AIRSCHEd::ReachableUniverse

[216](#)

AIRSCHEd::SegmentPathPeriod

[231](#)

AIRSCHEd::BomDisplay

[156](#)

CmdAbstract

[157](#)

AIRSCHEd::FlightPeriodFileParser

[179](#)

AIRSCHEd::InventoryGenerator

[188](#)

AIRSCHEd::OnDParser	194
AIRSCHEd::OnDPeriodFileParser	196
AIRSCHEd::OnDPeriodGenerator	197
AIRSCHEd::ScheduleParser	224
AIRSCHEd::SegmentPathGenerator	231
AIRSCHEd::SegmentPathProvider	243
AIRSCHEd::Simulator	248
AIRSCHEd::TravelSolutionParser	299
airsched::Date_T	158
AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >	160
airsched::SearchStringParser::definition< ScannerT >	164
AIRSCHEd::OnDParserHelper::OnDParser::definition< ScannerT >	167
AIRSCHEd::FacServiceAbstract	174
FacServiceAbstract	176
AIRSCHEd::FacAIRSCHEdServiceContext	173
FileNotFoundException	178
AIRSCHEd::OnDInputFileNotFoundException	193
AIRSCHEd::ScheduleInputFileNotFoundException	223
AIRSCHEd::FlagSaver	178
grammar	188
AIRSCHEd::OnDParserHelper::OnDParser	195
AIRSCHEd::ScheduleParserHelper::FlightPeriodParser	180
airsched::SearchStringParser	227
KeyAbstract	189
AIRSCHEd::OriginDestinationSetKey	207
AIRSCHEd::ReachableUniverseKey	221
AIRSCHEd::SegmentPathPeriodKey	238

ParserException	209
AIRSCHED::SegmentDateNotFoundExpection	230
AIRSCHED::OnDParserHelper::ParserSemanticAction	210
AIRSCHED::OnDParserHelper::doEndOnD	171
AIRSCHED::OnDParserHelper::storeAirlineCode	258
AIRSCHED::OnDParserHelper::storeClassCode	264
AIRSCHED::OnDParserHelper::storeDateRangeEnd	268
AIRSCHED::OnDParserHelper::storeDateRangeStart	271
AIRSCHED::OnDParserHelper::storeDestination	273
AIRSCHED::OnDParserHelper::storeEndRangeTime	277
AIRSCHED::OnDParserHelper::storeOrigin	289
AIRSCHED::OnDParserHelper::storeStartRangeTime	297
AIRSCHED::ScheduleParserHelper::ParserSemanticAction	211
AIRSCHED::ScheduleParserHelper::doEndFlight	169
AIRSCHED::ScheduleParserHelper::storeAirlineCode	260
AIRSCHED::ScheduleParserHelper::storeBoardingTime	261
AIRSCHED::ScheduleParserHelper::storeCapacity	263
AIRSCHED::ScheduleParserHelper::storeClasses	265
AIRSCHED::ScheduleParserHelper::storeDateRangeEnd	267
AIRSCHED::ScheduleParserHelper::storeDateRangeStart	270
AIRSCHED::ScheduleParserHelper::storeDow	274
AIRSCHED::ScheduleParserHelper::storeElapsedTime	276
AIRSCHED::ScheduleParserHelper::storeFamilyCode	279
AIRSCHED::ScheduleParserHelper::storeFClasses	280
AIRSCHED::ScheduleParserHelper::storeFlightNumber	282
AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint	283
AIRSCHED::ScheduleParserHelper::storeLegCabinCode	285

AIRSCHEd::ScheduleParserHelper::storeLegOffPoint	286
AIRSCHEd::ScheduleParserHelper::storeOffTime	288
AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint	291
AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode	292
AIRSCHEd::ScheduleParserHelper::storeSegmentOffPoint	294
AIRSCHEd::ScheduleParserHelper::storeSegmentSpecificity	295
airsched::Passenger_T	213
airsched::Place_T	215
airsched::SearchString_T	225
AIRSCHEd::SegmentPeriodHelper	244
ServiceAbstract	247
AIRSCHEd::AIRSCHEd_ServiceContext	155
AIRSCHEd::ServiceAbstract	247
airsched::store_adult_passenger_type	249
airsched::store_airline_code	250
airsched::store_airline_name	251
airsched::store_airline_sign	252
airsched::store_child_passenger_type	253
airsched::store_date	254
airsched::store_passenger_number	255
airsched::store_pet_passenger_type	256
airsched::store_place_element	257
StructAbstract	298
AIRSCHEd::FareFamilyStruct	176
AIRSCHEd::FlightPeriodStruct	181
AIRSCHEd::LegCabinStruct	189
AIRSCHEd::LegStruct	191

AIRSCHEd::OnDPeriodStruct	198
AIRSCHEd::SegmentCabinStruct	228
AIRSCHEd::SegmentStruct	245
TestFixture	298
AirlineScheduleTestSuite	149

21 Class Index

21.1 Class List

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

airsched::Airline_T	148
AirlineScheduleTestSuite	149
AIRSCHEd::AIRSCHEd_Service (Interface for the AirSched Services)	151
AIRSCHEd::AIRSCHEd_ServiceContext (Class holding the context of the AirSched services)	155
BomAbstract	156
AIRSCHEd::BomDisplay (Utility class to display AirSched objects with a pretty format)	156
CmdAbstract	157
airsched::Date_T	158
AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >	160
airsched::SearchStringParser::definition< ScannerT >	164
AIRSCHEd::OnDParserHelper::OnDParser::definition< ScannerT >	167
AIRSCHEd::ScheduleParserHelper::doEndFlight	169
AIRSCHEd::OnDParserHelper::doEndOnD	171
AIRSCHEd::FacAIRSCHEdServiceContext (Factory for the service context)	173
AIRSCHEd::FacServiceAbstract	174
FacServiceAbstract	176

AIRSCHED::FareFamilyStruct	176
FileNotFoundException	178
AIRSCHED::FlagSaver	178
AIRSCHED::FlightPeriodFileParser	179
AIRSCHED::ScheduleParserHelper::FlightPeriodParser	180
AIRSCHED::FlightPeriodStruct	181
grammar	188
AIRSCHED::InventoryGenerator	188
KeyAbstract	189
AIRSCHED::LegCabinStruct	189
AIRSCHED::LegStruct	191
AIRSCHED::OnDInputFileNotFoundException	193
AIRSCHED::OnDParser (Class wrapping the parser entry point)	194
AIRSCHED::OnDParserHelper::OnDParser	195
AIRSCHED::OnDPeriodFileParser	196
AIRSCHED::OnDPeriodGenerator (Class handling the generation / instantiation of the O&D-Period BOM)	197
AIRSCHED::OnDPeriodStruct	198
AIRSCHED::OriginDestinationSet (Class representing a simple sub-network)	202
AIRSCHED::OriginDestinationSetKey (Structure representing the key of a sub-network)	207
ParserException	209
AIRSCHED::OnDParserHelper::ParserSemanticAction	210
AIRSCHED::ScheduleParserHelper::ParserSemanticAction	211
airsched::Passenger_T	213
airsched::Place_T	215
AIRSCHED::ReachableUniverse (Class representing the root of the schedule-related BOM tree)	216

AIRSCHEd::ReachableUniverseKey (Structure representing the key of the schedule-related BOM tree root)	221
AIRSCHEd::ScheduleInputFileNotFoundException	223
AIRSCHEd::ScheduleParser	224
airsched::SearchString_T	225
airsched::SearchStringParser	227
AIRSCHEd::SegmentCabinStruct	228
AIRSCHEd::SegmentDateNotFoundException	230
AIRSCHEd::SegmentPathGenerator (Class handling the generation / instantiation of the network BOM)	231
AIRSCHEd::SegmentPathPeriod (Class representing a segment/path)	231
AIRSCHEd::SegmentPathPeriodKey (Structure representing the key of a segment/path)	238
AIRSCHEd::SegmentPathProvider (Class building the travel solutions from airline schedules)	243
AIRSCHEd::SegmentPeriodHelper	244
AIRSCHEd::SegmentStruct	245
ServiceAbstract	247
AIRSCHEd::ServiceAbstract	247
AIRSCHEd::Simulator	248
airsched::store_adult_passenger_type	249
airsched::store_airline_code	250
airsched::store_airline_name	251
airsched::store_airline_sign	252
airsched::store_child_passenger_type	253
airsched::store_date	254
airsched::store_passenger_number	255
airsched::store_pet_passenger_type	256
airsched::store_place_element	257

AIRSCHEd::OnDParserHelper::storeAirlineCode	258
AIRSCHEd::ScheduleParserHelper::storeAirlineCode	260
AIRSCHEd::ScheduleParserHelper::storeBoardingTime	261
AIRSCHEd::ScheduleParserHelper::storeCapacity	263
AIRSCHEd::OnDParserHelper::storeClassCode	264
AIRSCHEd::ScheduleParserHelper::storeClasses	265
AIRSCHEd::ScheduleParserHelper::storeDateRangeEnd	267
AIRSCHEd::OnDParserHelper::storeDateRangeEnd	268
AIRSCHEd::ScheduleParserHelper::storeDateRangeStart	270
AIRSCHEd::OnDParserHelper::storeDateRangeStart	271
AIRSCHEd::OnDParserHelper::storeDestination	273
AIRSCHEd::ScheduleParserHelper::storeDow	274
AIRSCHEd::ScheduleParserHelper::storeElapsedTime	276
AIRSCHEd::OnDParserHelper::storeEndRangeTime	277
AIRSCHEd::ScheduleParserHelper::storeFamilyCode	279
AIRSCHEd::ScheduleParserHelper::storeFCClasses	280
AIRSCHEd::ScheduleParserHelper::storeFlightNumber	282
AIRSCHEd::ScheduleParserHelper::storeLegBoardingPoint	283
AIRSCHEd::ScheduleParserHelper::storeLegCabinCode	285
AIRSCHEd::ScheduleParserHelper::storeLegOffPoint	286
AIRSCHEd::ScheduleParserHelper::storeOffTime	288
AIRSCHEd::OnDParserHelper::storeOrigin	289
AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint	291
AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode	292
AIRSCHEd::ScheduleParserHelper::storeSegmentOffPoint	294
AIRSCHEd::ScheduleParserHelper::storeSegmentSpecificity	295
AIRSCHEd::OnDParserHelper::storeStartRangeTime	297

StructAbstract	298
TestFixture	298
AIRSCHED::TravelSolutionParser (Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file)	299

22 File Index

22.1 File List

Here is a list of all files with brief descriptions:

airsched/AIRSCHED_Service.hpp	300
airsched/AIRSCHED_Types.hpp	302
airsched/basic/BasConst.cpp	303
airsched/basic/BasConst_AIRSCHED_Service.hpp	304
airsched/basic/BasConst_General.hpp	304
airsched/basic/BasParserTypes.hpp	306
airsched/batches/airsched.cpp	310
airsched/batches/BookingRequestParser.cpp	318
airsched/batches/BookingRequestParser.hpp	324
airsched/bom/AirportList.hpp	327
airsched/bom/BomDisplay.cpp	328
airsched/bom/BomDisplay.hpp	330
airsched/bom/FareFamilyStruct.cpp	330
airsched/bom/FareFamilyStruct.hpp	331
airsched/bom/FlightPeriodStruct.cpp	332
airsched/bom/FlightPeriodStruct.hpp	337
airsched/bom/LegCabinStruct.cpp	339
airsched/bom/LegCabinStruct.hpp	340
airsched/bom/LegStruct.cpp	340

airsched/bom/ LegStruct.hpp	342
airsched/bom/ OnDPeriodStruct.cpp	343
airsched/bom/ OnDPeriodStruct.hpp	345
airsched/bom/ OriginDestinationSet.cpp	346
airsched/bom/ OriginDestinationSet.hpp	348
airsched/bom/ OriginDestinationSetKey.cpp	351
airsched/bom/ OriginDestinationSetKey.hpp	352
airsched/bom/ OriginDestinationSetTypes.hpp	354
airsched/bom/ ReachableUniverse.cpp	355
airsched/bom/ ReachableUniverse.hpp	357
airsched/bom/ ReachableUniverseKey.cpp	359
airsched/bom/ ReachableUniverseKey.hpp	361
airsched/bom/ ReachableUniverseTypes.hpp	363
airsched/bom/ SegmentCabinStruct.cpp	364
airsched/bom/ SegmentCabinStruct.hpp	365
airsched/bom/ SegmentPathPeriod.cpp	366
airsched/bom/ SegmentPathPeriod.hpp	372
airsched/bom/ SegmentPathPeriodKey.cpp	375
airsched/bom/ SegmentPathPeriodKey.hpp	377
airsched/bom/ SegmentPathPeriodTypes.hpp	380
airsched/bom/ SegmentPeriodHelper.cpp	381
airsched/bom/ SegmentPeriodHelper.hpp	382
airsched/bom/ SegmentStruct.cpp	383
airsched/bom/ SegmentStruct.hpp	385
airsched/command/ InventoryGenerator.cpp	386
airsched/command/ InventoryGenerator.hpp	388
airsched/command/ OnDParser.cpp	389

airsched/command/OnDParser.hpp	390
airsched/command/OnDParserHelper.cpp	392
airsched/command/OnDParserHelper.hpp	399
airsched/command/OnDPeriodGenerator.cpp	402
airsched/command/OnDPeriodGenerator.hpp	403
airsched/command/ScheduleParser.cpp	404
airsched/command/ScheduleParser.hpp	405
airsched/command/ScheduleParserHelper.cpp	406
airsched/command/ScheduleParserHelper.hpp	419
airsched/command/SegmentPathGenerator.cpp	422
airsched/command/SegmentPathGenerator.hpp	430
airsched/command/SegmentPathProvider.cpp	431
airsched/command/SegmentPathProvider.hpp	434
airsched/command/Simulator.cpp	435
airsched/command/Simulator.hpp	436
airsched/command/TravelSolutionParser.cpp	437
airsched/command/TravelSolutionParser.hpp	440
airsched/config/airsched-paths.hpp	442
airsched/config/airsched-paths.hpp.in	442
airsched/factory/FacAIRSCHEDServiceContext.cpp	443
airsched/factory/FacAIRSCHEDServiceContext.hpp	444
airsched/factory/FacServiceAbstract.cpp	445
airsched/factory/FacServiceAbstract.hpp	446
airsched/service/AIRSCHED_Service.cpp	447
airsched/service/AIRSCHED_ServiceContext.cpp	453
airsched/service/AIRSCHED_ServiceContext.hpp	455
airsched/service/ServiceAbstract.cpp	456

airsched/service/ServiceAbstract.hpp	457
test/airsched/AirlineScheduleTestSuite.cpp	459
test/airsched/AirlineScheduleTestSuite.hpp	462

23 Directory Documentation

23.1 test/airsched/ Directory Reference

Files

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

23.2 aairsched/ Directory Reference

Directories

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

Files

- file [AIRSCHED_Service.hpp](#)
- file [AIRSCHED_Types.hpp](#)

23.3 aairsched/basic/ Directory Reference

Files

- file [BasConst.cpp](#)
- file [BasConst_AIRSCHED_Service.hpp](#)
- file [BasConst_General.hpp](#)
- file [BasParserTypes.hpp](#)

23.4 airsched/batches/ Directory Reference

Files

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

23.5 airsched/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 airsched/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 airsched/config/ Directory Reference

Files

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

23.8 airsched/factory/ Directory Reference

Files

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

23.9 airsched/service/ Directory Reference

Files

- file [AIRSCHED_Service.cpp](#)
- file [AIRSCHED_ServiceContext.cpp](#)
- file [AIRSCHED_ServiceContext.hpp](#)
- file [ServiceAbstract.cpp](#)
- file [ServiceAbstract.hpp](#)

23.10 test/ Directory Reference

Directories

- directory [airsched](#)

24 Namespace Documentation

24.1 airsched 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> airsched::PlaceList_T

List of Place strucutres.

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

24.1.1.2 typedef std::vector<Date_T> airsched::DateList_T

List of Date strucutres.

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

24.1.1.3 typedef std::vector<Airline_T> airsched::AirlineList_T

List of Airline strucutres.

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

24.1.1.4 typedef std::vector<Passenger_T> airsched::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 airsched::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 passengers ::=
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'

```

Definition at line 373 of file [BookingRequestParser.cpp](#).

24.1.3 Variable Documentation

24.1.3.1 `boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> airsched::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> airsched::uint1_p`

1-digit-integer parser

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

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

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

Up-to-2-digit-integer parser

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

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

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

2-digit-integer parser

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

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

24.1.3.5 `boost::spirit::classic::uint_parser<int, 10, 2, 4> airsched::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> airsched::uint4_p`

4-digit-integer parser

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

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

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

Up-to-4-digit-integer parser

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

24.2 AIRSCHED Namespace Reference

Namespaces

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

Classes

- class [AIRSCHED_Service](#)
Interface for the AirSched Services.
- class [SegmentDateNotFoundException](#)
- class [OnDInputFileNotFoundException](#)
- class [ScheduleInputFileNotFoundException](#)
- struct [FlagSaver](#)
- class [BomDisplay](#)
Utility class to display AirSched 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](#)
- 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 [FacAIRSCHEDServiceContext](#)

Factory for the service context.

- class [FacServiceAbstract](#)
- class [AIRSCHED_ServiceContext](#)

Class holding the context of the AirSched services.

- class [ServiceAbstract](#)

Typedefs

- typedef boost::shared_ptr< [AIRSCHED_Service](#) > [AIRSCHED_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

- 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)

Variables

- const int [DEFAULT_NUMBER_OF_DRAWNS_FOR_MC_SIMULATION](#) = 100000

24.2.1 Typedef Documentation

24.2.1.1 typedef boost::shared_ptr<[AIRSCHED_Service](#)>
[AIRSCHED::AIRSCHED_ServicePtr_T](#)

(Smart) Pointer on the AirSched service handler.

Definition at line 62 of file [AIRSCHED_Types.hpp](#).

24.2.1.2 typedef char AIRSCHED::char_t

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

24.2.1.3 typedef boost::spirit::classic::file_iterator<char_t> AIRSCHED::iterator_t

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

24.2.1.4 typedef boost::spirit::classic::scanner<iterator_t> AIRSCHED::scanner_t

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

24.2.1.5 typedef boost::spirit::classic::rule<scanner_t> AIRSCHED::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> AIRSCHED::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> AIRSCHED::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> AIRSCHED::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> AIRSCHED::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> AIRSCHED::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 AIRSCHED::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>
           AIRSCHED::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>
           AIRSCHED::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>
           AIRSCHED::bounded1_4_p_t
```

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

```
24.2.1.15  typedef std::set<stdair::AirportCode_T> AIRSCHED::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> AIRSCHED::AirportOrderedList_T
```

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

```
24.2.1.17  typedef std::vector<FareFamilyStruct>
           AIRSCHED::FareFamilyStructList_T
```

List of FareFamily-Detail structures.

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

```
24.2.1.18  typedef std::vector<LegCabinStruct> AIRSCHED::LegCabinStructList_T
```

List of LegCabin-Detail structures.

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

```
24.2.1.19  typedef std::vector<LegStruct> AIRSCHED::LegStructList_T
```

List of Leg structures.

Definition at line 50 of file [LegStruct.hpp](#).

```
24.2.1.20  typedef std::list<OriginDestinationSet*>
           AIRSCHED::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*>
AIRSCHED::OriginDestinationSetMap_T`

Define the [OriginDestinationSet](#) map.

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

24.2.1.22 `typedef std::list<ReachableUniverse*>
AIRSCHED::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*>
AIRSCHED::ReachableUniverseMap_T`

Define the reachable-universe map.

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

24.2.1.24 `typedef std::vector<SegmentCabinStruct>
AIRSCHED::SegmentCabinStructList_T`

List of SegmentCabin-Detail structures.

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

24.2.1.25 `typedef std::list<SegmentPathPeriod*>
AIRSCHED::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*>
AIRSCHED::SegmentPathPeriodMultimap_T`

Define the segment path map.

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

24.2.1.27 `typedef std::vector<const SegmentPathPeriod*>
AIRSCHED::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>
AIRSCHED::SegmentPathPeriodListList_T`

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

24.2.1.29 `typedef std::vector<stdair::DateOffset_T> AIRSCHED::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> AIRSCHED::SegmentStructList_T`

List of Segment strucutres.

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

24.2.2 Function Documentation

24.2.2.1 `template void AIRSCHED::OriginDestinationSet::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

24.2.2.2 `template void AIRSCHED::OriginDestinationSet::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

24.2.2.3 `template void AIRSCHED::OriginDestinationSetKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

24.2.2.4 `template void AIRSCHED::OriginDestinationSetKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

24.2.2.5 `template void AIRSCHED::ReachableUniverse::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

24.2.2.6 `template void AIRSCHED::ReachableUniverse::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

24.2.2.7 `template void AIRSCHED::ReachableUniverseKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

24.2.2.8 `template void AIRSCHED::ReachableUniverseKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

24.2.3 Variable Documentation

24.2.3.1 `const int AIRSCHED::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](#).

24.3 AIRSCHED::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 AIRSCHED::OnDParserHelper::alpha_cap_set_p](#) ("A-Z")

Sequence of (capital) alphabetic characters: [chset_p](#)("A-Z")

24.3.1.2 [repeat_p_t AIRSCHED::OnDParserHelper::airport_p](#) ([chset_t](#)("0-9A-Z").derived() , 3 , 3)

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

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

24.3.1.3 **repeat_p_t** AIRSCHED::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 [AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >::definition\(\)](#).

24.3.1.4 **bounded4_p_t** AIRSCHED::OnDParserHelper::year_p (*uint4_p. derived()*, 2000u, 2099u)

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

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

24.3.1.5 **bounded2_p_t** AIRSCHED::OnDParserHelper::month_p (*uint2_p. derived()*, 1u, 12u)

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

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

24.3.1.6 **bounded2_p_t** AIRSCHED::OnDParserHelper::day_p (*uint2_p. derived()*, 1u, 31u)

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

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

24.3.1.7 **bounded2_p_t** AIRSCHED::OnDParserHelper::hours_p (*uint2_p. derived()*, 0u, 23u)

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

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

24.3.1.8 **bounded2_p_t** AIRSCHED::OnDParserHelper::minutes_p (*uint2_p. derived()*, 0u, 59u)

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

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

24.3.1.9 **bounded2_p_t** AIRSCHED::OnDParserHelper::seconds_p (*uint2_p. derived()*, 0u, 59u)

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

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

24.3.1.10 **chset_t** AIRSCHED::OnDParserHelper::class_code_p ("A-Z")

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

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

24.3.2 Variable Documentation

24.3.2.1 uint2_p_t AIRSCHED::OnDParserHelper::uint2_p

2-digit-integer parser

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

24.3.2.2 uint4_p_t AIRSCHED::OnDParserHelper::uint4_p

4-digit-integer parser

Definition at line 218 of file [OnDParserHelper.cpp](#).

24.3.2.3 uint1_4_p_t AIRSCHED::OnDParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 221 of file [OnDParserHelper.cpp](#).

24.4 AIRSCHED::ScheduleParserHelper Namespace Reference

Classes

- struct [ParserSemanticAction](#)
- struct [storeAirlineCode](#)
- struct [storeFlightNumber](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeDow](#)
- struct [storeLegBoardingPoint](#)
- struct [storeLegOffPoint](#)
- struct [storeBoardingTime](#)
- struct [storeOffTime](#)
- struct [storeElapsedTime](#)
- struct [storeLegCabinCode](#)
- struct [storeCapacity](#)
- struct [storeSegmentSpecificity](#)
- struct [storeSegmentBoardingPoint](#)
- struct [storeSegmentOffPoint](#)
- struct [storeSegmentCabinCode](#)
- struct [storeClasses](#)
- struct [storeFamilyCode](#)
- struct [storeFClasses](#)
- struct [doEndFlight](#)
- struct [FlightPeriodParser](#)

Functions

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

Variables

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

24.4.1 Function Documentation

24.4.1.1 **repeat_p_t** AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

24.4.1.2 **bounded1_4_p_t** AIRSCHED::ScheduleParserHelper::flight_number_p (uint1_4_p.
derived(), 0u, 9999u)

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

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

24.4.1.3 **bounded4_p_t** AIRSCHED::ScheduleParserHelper::year_p (uint4_p.
derived(), 2000u, 2099u)

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

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

24.4.1.4 **bounded2_p_t** AIRSCHED::ScheduleParserHelper::month_p (uint2_p. *derived()*, 1u, 12u)

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

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

24.4.1.5 **bounded2_p_t** AIRSCHED::ScheduleParserHelper::day_p (uint2_p. *derived()*, 1u, 31u)

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

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

24.4.1.6 **repeat_p_t** AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

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

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

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

24.4.1.8 **bounded2_p_t** AIRSCHED::ScheduleParserHelper::hours_p (uint2_p. *derived()*, 0u, 23u)

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

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

24.4.1.9 **bounded2_p_t** AIRSCHED::ScheduleParserHelper::minutes_p (uint2_p. *derived()*, 0u, 59u)

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

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

24.4.1.10 **bounded2_p_t** AIRSCHED::ScheduleParserHelper::seconds_p (uint2_p. *derived()*, 0u, 59u)

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

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

24.4.1.11 chset_t AIRSCHED::ScheduleParserHelper::cabin_code_p ("A-Z")

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

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

24.4.1.12 repeat_p_t AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition\(\)](#).

24.4.2 Variable Documentation

24.4.2.1 int1_p_t AIRSCHED::ScheduleParserHelper::int1_p

1-digit-integer parser

Definition at line 405 of file [ScheduleParserHelper.cpp](#).

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

24.4.2.2 uint2_p_t AIRSCHED::ScheduleParserHelper::uint2_p

2-digit-integer parser

Definition at line 408 of file [ScheduleParserHelper.cpp](#).

24.4.2.3 uint4_p_t AIRSCHED::ScheduleParserHelper::uint4_p

4-digit-integer parser

Definition at line 411 of file [ScheduleParserHelper.cpp](#).

24.4.2.4 uint1_4_p_t AIRSCHED::ScheduleParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 414 of file [ScheduleParserHelper.cpp](#).

24.4.2.5 int1_p_t AIRSCHED::ScheduleParserHelper::family_code_p

Family code parser

Definition at line 450 of file [ScheduleParserHelper.cpp](#).

Referenced by [AIRSCHED::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 airsched::Airline_T Struct Reference

```
#include <airsched/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 `airsched::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 aircshed::Airline_T::display () const` `[inline]`

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

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

Referenced by [airsched::SearchString_T::display\(\)](#).

25.1.4 Member Data Documentation

25.1.4.1 `bool aircshed::Airline_T::_isPreferred`

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

Referenced by [display\(\)](#), and [airsched::store_airline_sign::operator\(\)](#).

25.1.4.2 `std::string aircshed::Airline_T::_name`

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

Referenced by [display\(\)](#), and [airsched::store_airline_name::operator\(\)](#).

25.1.4.3 `std::string aircshed::Airline_T::_code`

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

Referenced by [display\(\)](#), and [airsched::store_airline_code::operator\(\)](#).

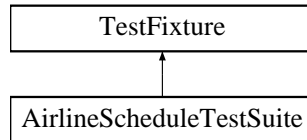
The documentation for this struct was generated from the following file:

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

25.2 AirlineScheduleTestSuite Class Reference

```
#include <test/airsched/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/airsched/[AirlineScheduleTestSuite.hpp](#)

25.3 AIRSCHED::AIRSCHED_Service Class Reference

Interface for the AirSched Services.

```
#include <airsched/AIRSCHED_Service.hpp>
```

Public Member Functions

- [AIRSCHED_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [AIRSCHED_Service](#) (const stdair::BasLogParams &)
- [AIRSCHED_Service](#) (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr)
- void [parseAndLoad](#) (const stdair::Filename_T &iScheduleInputFilename)
- void [parseAndLoad](#) (const stdair::Filename_T &iScheduleFilename, const stdair::Filename_T &iODInputFilename)
- [~AIRSCHED_Service](#) ()
- void [buildSampleBom](#) ()
- void [buildSegmentPathList](#) (stdair::TravelSolutionList_T &, const stdair::BookingRequestStruct &)
- void [simulate](#) ()
- std::string [jsonExport](#) (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 AirSched Services.

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

25.3.2 Constructor & Destructor Documentation

25.3.2.1 AIRSCHED::AIRSCHED_Service::AIRSCHED_Service (const stdair::BasLogParams & iLogParams, const stdair::BasDBParams & iDBParams)

Constructor.

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

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

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

Parameters

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

Definition at line 62 of file [AIRSCHED_Service.cpp](#).

25.3.2.2 AIRSCHED::AIRSCHED_Service::AIRSCHED_Service (*const* stdair::BasLogParams & *iLogParams*)

Constructor.

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

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

Parameters

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

Definition at line 42 of file [AIRSCHED_Service.cpp](#).

25.3.2.3 AIRSCHED::AIRSCHED_Service::AIRSCHED_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_ServicePtr*)

Constructor.

The `initAirschedService()` 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 [AIRSCHED_Service](#) is itself being initialised by another library service such as `SIMCRS_Service`).

Parameters

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

Definition at line 84 of file [AIRSCHED_Service.cpp](#).

25.3.2.4 AIRSCHED::AIRSCHED_Service::~~AIRSCHED_Service ()

Destructor.

Definition at line 100 of file [AIRSCHED_Service.cpp](#).

25.3.3 Member Function Documentation

25.3.3.1 `void AIRSCHED::AIRSCHED_Service::parseAndLoad (const stdair::Filename_T & iScheduleInputFilename)`

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

<code>const stdair::Filename_T</code> & Filename of the input schedule file.
--

Definition at line 178 of file [AIRSCHED_Service.cpp](#).

References [AIRSCHED::ScheduleParser::generateInventories\(\)](#).

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

25.3.3.2 `void AIRSCHED::AIRSCHED_Service::parseAndLoad (const stdair::Filename_T & iScheduleFilename, const stdair::Filename_T & iODInputFilename)`

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

<code>const stdair::Filename_T</code> & Filename of the input schedule file.
--

<code>const stdair::Filename_T</code> & Filename of the input O&D file.

Definition at line 199 of file [AIRSCHED_Service.cpp](#).

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

25.3.3.3 `void AIRSCHED::AIRSCHED_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 223 of file [AIRSCHED_Service.cpp](#).

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

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

25.3.3.4 `void AIRSCHED::AIRSCHED_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 369 of file [AIRSCHED_Service.cpp](#).

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

25.3.3.5 void AIRSCHED::AIRSCHED_Service::simulate ()

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

Currently, that method does nothing.

Definition at line 341 of file [AIRSCHED_Service.cpp](#).

25.3.3.6 std::string AIRSCHED::AIRSCHED_Service::jsonExport (const stdair::AirlineCode_T & *iAirlineCode*, const stdair::FlightNumber_T & *iFlightNumber*, const stdair::Date_T & *iDepartureDate*) const

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

Parameters

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

Returns

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

Definition at line 274 of file [AIRSCHED_Service.cpp](#).

25.3.3.7 std::string AIRSCHED::AIRSCHED_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 297 of file [AIRSCHED_Service.cpp](#).

25.3.3.8 std::string AIRSCHED::AIRSCHED_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

<i>const</i>	stdair::AirlineCode_T & Airline code of the flight period to display.
<i>const</i>	stdair::FlightNumber_T & Flight number of the flight to display.
<i>const</i>	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 318 of file [AIRSCHED_Service.cpp](#).

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

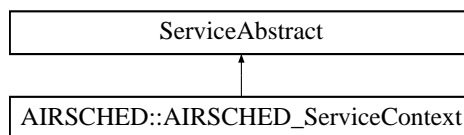
- [airsched/AIRSCHED_Service.hpp](#)
- [airsched/service/AIRSCHED_Service.cpp](#)

25.4 AIRSCHED::AIRSCHED_ServiceContext Class Reference

Class holding the context of the AirSched services.

```
#include <airsched/service/AIRSCHED_ServiceContext.hpp>
```

Inheritance diagram for AIRSCHED::AIRSCHED_ServiceContext:



Friends

- class [AIRSCHED_Service](#)
- class [FacAIRSCHEDServiceContext](#)

25.4.1 Detailed Description

Class holding the context of the AirSched services.

Definition at line 22 of file [AIRSCHED_ServiceContext.hpp](#).

25.4.2 Friends And Related Function Documentation

25.4.2.1 friend class AIRSCHED_Service [friend]

The [AIRSCHED_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 [AIRSCHED_ServiceContext.hpp](#).

25.4.2.2 friend class FacAIRSCHEDServiceContext [friend]

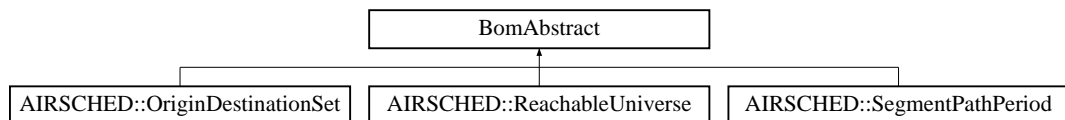
Definition at line 29 of file [AIRSCHED_ServiceContext.hpp](#).

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

- [airsched/service/AIRSCHED_ServiceContext.hpp](#)
- [airsched/service/AIRSCHED_ServiceContext.cpp](#)

25.5 BomAbstract Class Reference

Inheritance diagram for BomAbstract:



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

- [airsched/bom/OriginDestinationSet.hpp](#)

25.6 AIRSCHED::BomDisplay Class Reference

Utility class to display AirSched objects with a pretty format.

```
#include <airsched/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 AirSched 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 AIRSCHED::BomDisplay::csvDisplay (const stdair::BomRoot & iBomRoot)`
[static]

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

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> stdair::EventQueue&	Root of the BOM tree to be displayed.

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

25.6.2.2 void AIRSCHED::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

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> ReachableUniverse&	Root of the BOM tree to be displayed.

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

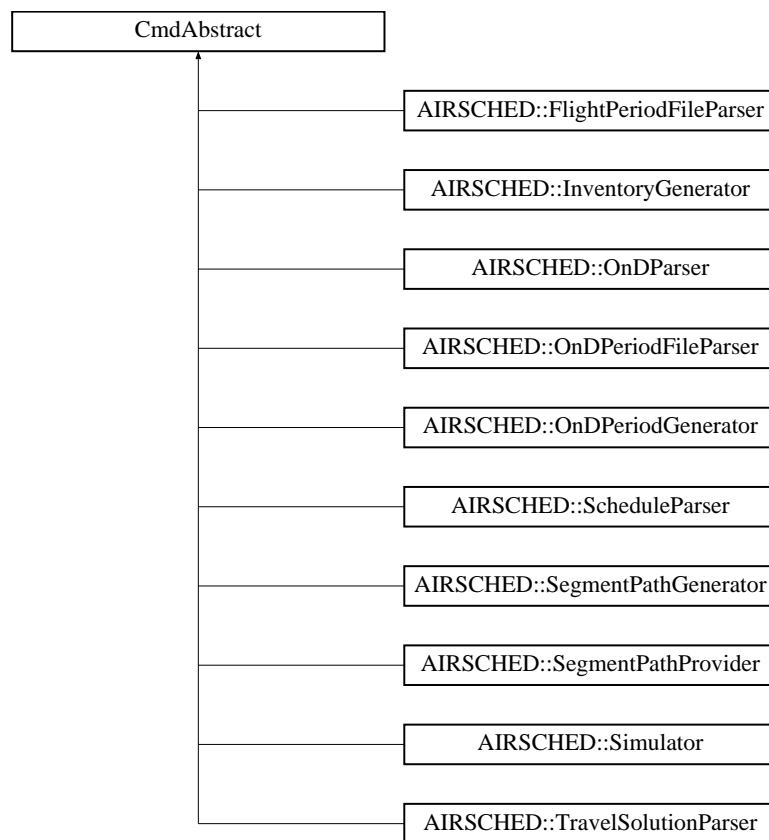
References [AIRSCHED::ReachableUniverse::toString\(\)](#).

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

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

25.7 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



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

- airsched/command/[OnDPeriodGenerator.hpp](#)

25.8 airsched::Date_T Struct Reference

```
#include <airsched/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 `airsched::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 airsched::Date_T::display () const` `[inline]`

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

References `_date`, `_day`, `_month`, `_reldays`, and `_year`.

Referenced by [airsched::SearchString_T::display\(\)](#).

25.8.3.2 `boost::gregorian::date airsched::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 [airsched::store_date::operator\(\)\(\)](#).

25.8.4 Member Data Documentation

25.8.4.1 `boost::gregorian::date airsched::Date_T::_date`

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

Referenced by [display\(\)](#), and [airsched::store_date::operator\(\)\(\)](#).

25.8.4.2 `unsigned int airsched::Date_T::_reldays`

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

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

25.8.4.3 unsigned int airtched::Date_T::_day

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

Referenced by [display\(\)](#), and [getDate\(\)](#).

25.8.4.4 unsigned int airtched::Date_T::_month

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

Referenced by [display\(\)](#), and [getDate\(\)](#).

25.8.4.5 unsigned int airtched::Date_T::_year

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

Referenced by [display\(\)](#), and [getDate\(\)](#).

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

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

25.9 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference

```
#include <airched/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 > flight_period_end](#)
- [boost::spirit::classic::rule< ScannerT > flight_key](#)
- [boost::spirit::classic::rule< ScannerT > airline_code](#)
- [boost::spirit::classic::rule< ScannerT > flight_number](#)
- [boost::spirit::classic::rule< ScannerT > date](#)
- [boost::spirit::classic::rule< ScannerT > dow](#)
- [boost::spirit::classic::rule< ScannerT > time](#)
- [boost::spirit::classic::rule< ScannerT > date_offset](#)
- [boost::spirit::classic::rule< ScannerT > leg](#)
- [boost::spirit::classic::rule< ScannerT > leg_key](#)
- [boost::spirit::classic::rule< ScannerT > leg_details](#)
- [boost::spirit::classic::rule< ScannerT > leg_cabin_details](#)
- [boost::spirit::classic::rule< ScannerT > segment_section](#)

- `boost::spirit::classic::rule< ScannerT >` [segment_key](#)
- `boost::spirit::classic::rule< ScannerT >` [full_segment_cabin_details](#)
- `boost::spirit::classic::rule< ScannerT >` [segment_cabin_details](#)
- `boost::spirit::classic::rule< ScannerT >` [full_family_cabin_details](#)
- `boost::spirit::classic::rule< ScannerT >` [family_cabin_details](#)
- `boost::spirit::classic::rule< ScannerT >` [generic_segment](#)
- `boost::spirit::classic::rule< ScannerT >` [specific_segment_list](#)

25.9.1 Detailed Description

`template<typename ScannerT>struct AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >`

Definition at line 256 of file [ScheduleParserHelper.hpp](#).

25.9.2 Constructor & Destructor Documentation

25.9.2.1 `template<typename ScannerT > AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition (FlightPeriodParser const & self)`

Definition at line 471 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::ScheduleParserHelper::airline_code_p\(\)](#), [AIRSCHED::ScheduleParserHelper::airport_p\(\)](#), [AIRSCHED::ScheduleParserHelper::cabin_code_p\(\)](#), [AIRSCHED::ScheduleParserHelper::class_code_list_p\(\)](#), [AIRSCHED::ScheduleParserHelper::day_p\(\)](#), [AIRSCHED::ScheduleParserHelper::dow_p\(\)](#), [AIRSCHED::ScheduleParserHelper::family_code_p](#), [AIRSCHED::ScheduleParserHelper::flight_number_p\(\)](#), [AIRSCHED::ScheduleParserHelper::hours_p\(\)](#), [AIRSCHED::ScheduleParserHelper::int1_p](#), [AIRSCHED::ScheduleParserHelper::minutes_p\(\)](#), [AIRSCHED::ScheduleParserHelper::month_p\(\)](#), [AIRSCHED::ScheduleParserHelper::seconds_p\(\)](#), and [AIRSCHED::ScheduleParserHelper::year_p\(\)](#).

25.9.3 Member Function Documentation

25.9.3.1 `template<typename ScannerT > boost::spirit::classic::rule< ScannerT > const & AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::start () const`

Entry point of the parser.

Definition at line 622 of file [ScheduleParserHelper.cpp](#).

25.9.4 Member Data Documentation

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

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

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

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.3 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::flight_period_end`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.4 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::flight_key`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.5 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::airline_code`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.6 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::flight_number`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.7 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::date`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.8 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::dow`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.9 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::time`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.10 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::date_offset`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.11 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::leg`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.12 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::leg_key`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.13 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::leg_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.14 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::leg_cabin_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.15 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::segment_section`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.16 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::segment_key`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.17 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT >::full_segment_cabin_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.18 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT>::segment_cabin_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.19 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT>::full_family_cabin_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.20 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT>::family_cabin_details`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.21 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT>::generic_segment`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

25.9.4.22 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition<
ScannerT>::specific_segment_list`

Definition at line 260 of file [ScheduleParserHelper.hpp](#).

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

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

25.10 airsched::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 airsched::SearchStringParser::definition< ScannerT >`

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

25.10.2 Constructor & Destructor Documentation

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

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

References [airsched::uint1_2_p](#), [airsched::uint1_p](#), [airsched::uint2_p](#), and [airsched::uint4_p](#).

25.10.3 Member Function Documentation

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

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

25.10.4 Member Data Documentation

25.10.4.1 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT> airsched::SearchStringParser::definition< ScannerT >::search_string`

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

25.10.4.2 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::places`

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

25.10.4.3 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::place_element`

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

25.10.4.4 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::dates`

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

25.10.4.5 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::date`

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

25.10.4.6 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::month`

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

25.10.4.7 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::day`

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

25.10.4.8 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::year`

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

25.10.4.9 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT
>::preferred_airlines`

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

25.10.4.10 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::airline_code`

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

25.10.4.11 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::airline_name`

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

25.10.4.12 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT >::passengers`

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

25.10.4.13 `template<typename ScannerT > boost::spirit::classic::rule<ScannerT>
airsched::SearchStringParser::definition< ScannerT
>::passenger_number`

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

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

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

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

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

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

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

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

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

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

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

25.11 AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT > Struct Template Reference

```
#include <airsched/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 AIRSCHED::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 > AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >::definition (OnDParser const & self)

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

References [AIRSCHED::OnDParserHelper::airline_code_p\(\)](#), [AIRSCHED::OnDParserHelper::airport_p\(\)](#), [AIRSCHED::OnDParserHelper::class_code_p\(\)](#), [AIRSCHED::OnDParserHelper::day_p\(\)](#), [AIRSCHED::OnDParserHelper::hours_p\(\)](#), [AIRSCHED::OnDParserHelper::minutes_p\(\)](#), [AIRSCHED::OnDParserHelper::month_p\(\)](#), [AIRSCHED::OnDParserHelper::seconds_p\(\)](#), and [AIRSCHED::OnDParserHelper::year_p\(\)](#).

25.11.3 Member Function Documentation

25.11.3.1 template<typename ScannerT > boost::spirit::classic::rule< ScannerT > const & AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >::start () const

Entry point of the parser.

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

25.11.4 Member Data Documentation

25.11.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >::ond_list

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

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

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

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

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

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

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

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

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

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

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

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

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

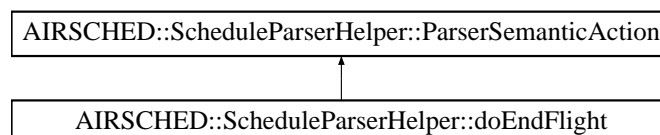
The documentation for this struct was generated from the following files:

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

25.12 AIRSCHED::ScheduleParserHelper::doEndFlight Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 193 of file [ScheduleParserHelper.hpp](#).

25.12.2 Constructor & Destructor Documentation

25.12.2.1 AIRSCHED::ScheduleParserHelper::doEndFlight::doEndFlight (stdair::BomRoot & *ioBomRoot*, [FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 372 of file [ScheduleParserHelper.cpp](#).

25.12.3 Member Function Documentation

25.12.3.1 void AIRSCHED::ScheduleParserHelper::doEndFlight::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

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

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

25.12.4 Member Data Documentation

25.12.4.1 stdair::BomRoot& AIRSCHED::ScheduleParserHelper::doEndFlight::_bomRoot

Actor Specific Context.

Definition at line 199 of file [ScheduleParserHelper.hpp](#).

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

25.12.4.2 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_- flightPeriod [inherited]

Actor Context.

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

Referenced by [operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirline::operator\(\)](#).

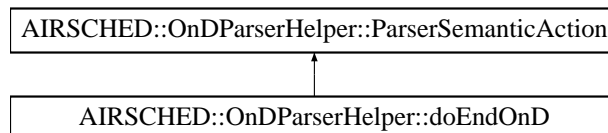
The documentation for this struct was generated from the following files:

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

25.13 AIRSCHED::OnDParserHelper::doEndOnD Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::doEndOnD:



Public Member Functions

- [doEndOnD](#) (stdair::BomRoot &, [OnDPeriodStruct](#) &)
- [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 AIRSCHED::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 AIRSCHED::OnDParserHelper::doEndOnD::operator() (*iterator_t iStr*, *iterator_t iStrEnd*) const

Actor Function (functor).

Definition at line 199 of file [OnDParserHelper.cpp](#).

References [_bomRoot](#), and [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#).

25.13.4 Member Data Documentation

25.13.4.1 *stdair::BomRoot&* AIRSCHED::OnDParserHelper::doEndOnD::_bomRoot

Actor Specific Context.

Definition at line 112 of file [OnDParserHelper.hpp](#).

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

25.13.4.2 *OnDPeriodStruct&* AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [operator\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#) and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

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

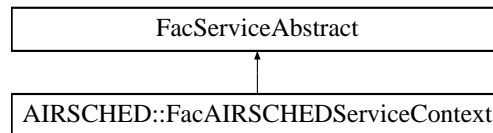
- [airsched/command/OnDParserHelper.hpp](#)
- [airsched/command/OnDParserHelper.cpp](#)

25.14 AIRSCHED::FacAIRSCHEDServiceContext Class Reference

Factory for the service context.

```
#include <airsched/factory/FacAIRSCHEDServiceContext.hpp>
```

Inheritance diagram for AIRSCHED::FacAIRSCHEDServiceContext:



Public Member Functions

- [~FacAIRSCHEDServiceContext \(\)](#)
- [AIRSCHED_ServiceContext & create \(\)](#)

Static Public Member Functions

- static [FacAIRSCHEDServiceContext & instance \(\)](#)

Protected Member Functions

- [FacAIRSCHEDServiceContext \(\)](#)

25.14.1 Detailed Description

Factory for the service context.

Definition at line 19 of file [FacAIRSCHEDServiceContext.hpp](#).

25.14.2 Constructor & Destructor Documentation

25.14.2.1 AIRSCHED::FacAIRSCHEDServiceContext::~~FacAIRSCHEDServiceContext ()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacAIRSCHED-ServiceContext::instance\(\)](#).

Definition at line 17 of file [FacAIRSCHEDServiceContext.cpp](#).

25.14.2.2 AIRSCHED::FacAIRSCHEDServiceContext::FacAIRSCHEDServiceContext () [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

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

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

25.14.3 Member Function Documentation

25.14.3.1 FacAIRSCHEDServiceContext & AIRSCHED::FacAIRSCHEDServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns

FacServiceContext&

Definition at line 22 of file [FacAIRSCHEDServiceContext.cpp](#).

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

25.14.3.2 AIRSCHED_ServiceContext & AIRSCHED::FacAIRSCHEDServiceContext::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 [FacAIRSCHEDServiceContext.cpp](#).

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

- [airsched/factory/FacAIRSCHEDServiceContext.hpp](#)
- [airsched/factory/FacAIRSCHEDServiceContext.cpp](#)

25.15 AIRSCHED::FacServiceAbstract Class Reference

```
#include <airsched/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*>
AIRSCHED::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 `AIRSCHED::FacServiceAbstract::~~FacServiceAbstract () [virtual]`

Destructor.

Definition at line 13 of file [FacServiceAbstract.cpp](#).

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

25.15.3.2 `AIRSCHED::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 AIRSCHED::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 AIRSCHED::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

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

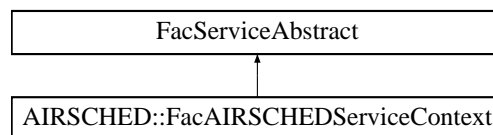
Referenced by [clean\(\)](#).

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

- [airsched/factory/FacServiceAbstract.hpp](#)
- [airsched/factory/FacServiceAbstract.cpp](#)

25.16 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



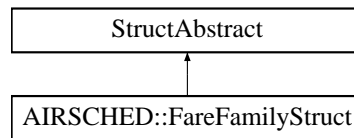
The documentation for this class was generated from the following file:

- [airsched/factory/FacAIRSCHEDServiceContext.hpp](#)

25.17 AIRSCHED::FareFamilyStruct Struct Reference

```
#include <airsched/bom/FareFamilyStruct.hpp>
```

Inheritance diagram for AIRSCHED::FareFamilyStruct:



Public Member Functions

- [FareFamilyStruct](#) (const stdair::FamilyCode_T &, const stdair::ClassList_String_T &)
- const std::string [describe](#) () const

Public Attributes

- stdair::FamilyCode_T [_familyCode](#)
- 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 [AIRSCHED::FareFamilyStruct::FareFamilyStruct](#) (const stdair::FamilyCode_T & *iFamilyCode*, 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 [AIRSCHED::FareFamilyStruct::describe](#) () const

Give a description of the structure (for display purposes).

Definition at line 21 of file [FareFamilyStruct.cpp](#).

References [_classes](#), and [_familyCode](#).

25.17.4 Member Data Documentation

25.17.4.1 stdair::FamilyCode_T [AIRSCHED::FareFamilyStruct::_familyCode](#)

Definition at line 19 of file [FareFamilyStruct.hpp](#).

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

25.17.4.2 stdair::ClassList_String_T AIRSCHED::FareFamilyStruct::_classes

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

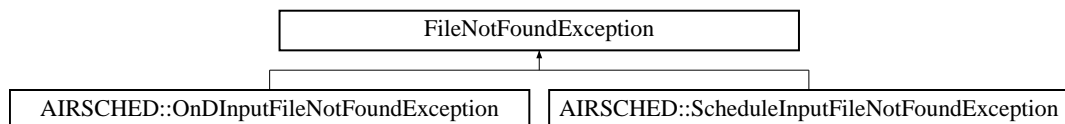
Referenced by [describe\(\)](#).

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

- [airsched/bom/FareFamilyStruct.hpp](#)
- [airsched/bom/FareFamilyStruct.cpp](#)

25.18 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException:



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

- [airsched/AIRSCHED_Types.hpp](#)

25.19 AIRSCHED::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 AIRSCHED::FlagSaver::FlagSaver (std::ostream & oStream) [inline]

Constructor.

Definition at line 25 of file [BomDisplay.cpp](#).

25.19.2.2 AIRSCHED::FlagSaver::~~FlagSaver () [inline]

Destructor.

Definition at line 30 of file [BomDisplay.cpp](#).

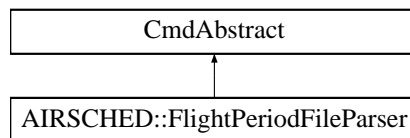
The documentation for this struct was generated from the following file:

- [airsched/bom/BomDisplay.cpp](#)

25.20 AIRSCHED::FlightPeriodFileParser Class Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 292 of file [ScheduleParserHelper.hpp](#).

25.20.2 Constructor & Destructor Documentation

25.20.2.1 AIRSCHED::FlightPeriodFileParser::FlightPeriodFileParser (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)

Constructor.

Definition at line 637 of file [ScheduleParserHelper.cpp](#).

25.21 AIRSCHED::ScheduleParserHelper::FlightPeriodParser Struct Reference

25.20.3 Member Function Documentation

25.20.3.1 bool AIRSCHED::FlightPeriodFileParser::generateInventories ()

Parse the input file and generate the Inventories.

Definition at line 674 of file [ScheduleParserHelper.cpp](#).

Referenced by [AIRSCHED::ScheduleParser::generateInventories\(\)](#).

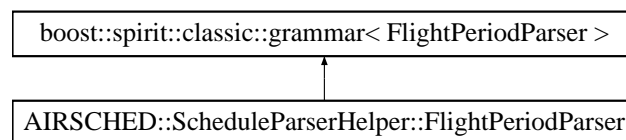
The documentation for this class was generated from the following files:

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

25.21 AIRSCHED::ScheduleParserHelper::FlightPeriodParser Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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; SegmentSpecificty (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; CDI; 1; CDI; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ SIN; SYD; C; CDI; 1; CDI; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ LHR; SYD; C; CDI; 1; CDI; M; YHBKLMNOPQRSTUVWXYZ; 2;YHBKLMNOPQRSTUVWXYZ

Grammar: DOW ::= int FlightKey ::= AirlineCode ';' FlightNumber ';' DateRangeStart ';' DateRangeEnd ';' DOW LegKey ::= BoardingPoint ';' OffPoint LegDetails ::= BoardingTime ['/ BoardingDateOffset] ';' OffTime ['/ BoardingDateOffset] ';' Elapsed Leg-CabinDetails ::= CabinCode ';' Capacity Leg ::= LegKey ';' LegDetails (';' CabinDetails)+ SegmentKey ::= BoardingPoint ';' OffPoint SegmentCabinDetails ::= CabinCode ';' Classes (';' FamilyCabinDetails)* FamilyCabinDetails ::= FamilyCode ';' Classes FullSegmentCabinDetails ::= (';' SegmentCabinDetails)+ GenericSegment ::= '0' (';' SegmentCabinDetails)+ SpecificSegments ::= '1' (';' SegmentKey ';' FullSegmentCabinDetails)+ SegmentSection ::= GenericSegment | SpecificSegments FlightPeriod ::= FlightKey (';' Leg)+ ';' SegmentSection ';' EndOfFlight EndOfFlight ::= ';' Grammar for the Flight-Period parser.

Definition at line 250 of file [ScheduleParserHelper.hpp](#).

25.21.2 Constructor & Destructor Documentation

25.21.2.1 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::FlightPeriodParser (stdair::BomRoot & ioBomRoot, FlightPeriodStruct & ioFlightPeriod)

Definition at line 462 of file [ScheduleParserHelper.cpp](#).

25.21.3 Member Data Documentation

25.21.3.1 stdair::BomRoot& AIRSCHED::ScheduleParserHelper::FlightPeriodParser::_bomRoot

Definition at line 273 of file [ScheduleParserHelper.hpp](#).

25.21.3.2 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::FlightPeriodParser::_flightPeriod

Definition at line 274 of file [ScheduleParserHelper.hpp](#).

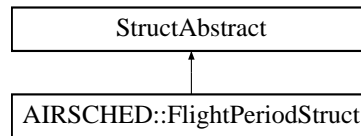
The documentation for this struct was generated from the following files:

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

25.22 AIRSCHED::FlightPeriodStruct Struct Reference

```
#include <airsched/bom/FlightPeriodStruct.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::FlightPeriodStruct::FlightPeriodStruct ()

Constructor.

Definition at line 17 of file [FlightPeriodStruct.cpp](#).

25.22.3 Member Function Documentation

25.22.3.1 stdair::Date_T AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#).

25.22.3.2 stdair::Duration_T AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#).

25.22.3.3 const std::string AIRSCHED::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](#), [AIRSCHED::SegmentStruct::describe\(\)](#), and [AIRSCHED::LegStruct::describe\(\)](#).

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#).

25.22.3.4 void AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

25.22.3.5 void AIRSCHED::FlightPeriodStruct::buildSegments ()

Build the list of [SegmentStruct](#) objects.

Definition at line 78 of file [FlightPeriodStruct.cpp](#).

References [_airportList](#), [_airportOrderedList](#), [AIRSCHED::SegmentStruct::_boardingPoint](#), [AIRSCHED::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#).

25.22.3.6 void AIRSCHED::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 [AIRSCHED::SegmentStruct::_boardingPoint](#), [AIRSCHED::SegmentStruct::_cabinList](#), [AIRSCHED::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#).

25.22.3.7 void AIRSCHED::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 [AIRSCHED::SegmentStruct::_cabinList](#), and [_segmentList](#).

25.22.3.8 void AIRSCHED::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 [AIRSCHED::SegmentStruct::_boardingPoint](#), [AIRSCHED::SegmentCabinStruct::_](#)

[cabinCode](#), [AIRSCHED::SegmentStruct::_cabinList](#), [AIRSCHED::SegmentCabinStruct::_fareFamilies](#), [AIRSCHED::SegmentStruct::_offPoint](#), and [_segmentList](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeFClasses::operator\(\)](#).

25.22.3.9 void AIRSCHED::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 [AIRSCHED::SegmentCabinStruct::_cabinCode](#), [AIRSCHED::SegmentStruct::_cabinList](#), [AIRSCHED::SegmentCabinStruct::_fareFamilies](#), and [_segmentList](#).

25.22.4 Member Data Documentation

25.22.4.1 stdair::AirlineCode_T AIRSCHED::FlightPeriodStruct::_airlineCode

Definition at line 84 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

25.22.4.2 stdair::FlightNumber_T AIRSCHED::FlightPeriodStruct::_flightNumber

Definition at line 85 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#).

25.22.4.3 stdair::DatePeriod_T AIRSCHED::FlightPeriodStruct::_dateRange

Definition at line 86 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

25.22.4.4 stdair::DoWStruct AIRSCHED::FlightPeriodStruct::_dow

Definition at line 87 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#).

25.22.4.5 LegStructList_T AIRSCHED::FlightPeriodStruct::_legList

Definition at line 88 of file [FlightPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegEndPoint::operator\(\)](#).

25.22.4.6 SegmentStructList_T AIRSCHED::FlightPeriodStruct::_segmentList

Definition at line 89 of file [FlightPeriodStruct.hpp](#).

Referenced by [addFareFamily\(\)](#), [addSegmentCabin\(\)](#), [buildSegments\(\)](#), and [describe\(\)](#).

25.22.4.7 bool AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::doEndCabin::operator\(\)](#).

25.22.4.8 LegStruct AIRSCHED::FlightPeriodStruct::_itLeg

Definition at line 94 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

25.22.4.9 LegCabinStruct AIRSCHED::FlightPeriodStruct::_itLegCabin

Definition at line 95 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeCapacityCabin::operator\(\)](#).

25.22.4.10 std::date AIRSCHED::FlightPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 98 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#).

25.22.4.11 std::date AIRSCHED::FlightPeriodStruct::_dateRangeEnd

Definition at line 99 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#).

25.22.4.12 unsigned int AIRSCHED::FlightPeriodStruct::_itYear

Definition at line 100 of file [FlightPeriodStruct.hpp](#).

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

25.22.4.13 unsigned int AIRSCHED::FlightPeriodStruct::_itMonth

Definition at line 101 of file [FlightPeriodStruct.hpp](#).

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

25.22.4.14 unsigned int AIRSCHED::FlightPeriodStruct::_itDay

Definition at line 102 of file [FlightPeriodStruct.hpp](#).

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

25.22.4.15 int AIRSCHED::FlightPeriodStruct::_dateOffset

Definition at line 103 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#).

25.22.4.16 long AIRSCHED::FlightPeriodStruct::_itHours

Staging Time.

Definition at line 106 of file [FlightPeriodStruct.hpp](#).

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

25.22.4.17 long AIRSCHED::FlightPeriodStruct::_itMinutes

Definition at line 107 of file [FlightPeriodStruct.hpp](#).

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

25.22.4.18 long AIRSCHED::FlightPeriodStruct::_itSeconds

Definition at line 108 of file [FlightPeriodStruct.hpp](#).

Referenced by [getTime\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#).

25.22.4.19 AirportList_T AIRSCHED::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 [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#).

25.22.4.20 AirportOrderedList_T AIRSCHED::FlightPeriodStruct::_airportOrderedList

Definition at line 113 of file [FlightPeriodStruct.hpp](#).

Referenced by [addAirport\(\)](#), [buildSegments\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#).

25.22.4.21 bool AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 116 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeFCClasses::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#).

25.22.4.22 SegmentStruct AIRSCHED::FlightPeriodStruct::_itSegment

Definition at line 117 of file [FlightPeriodStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeFClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentCabin](#)

25.22.4.23 SegmentCabinStruct AIRSCHED::FlightPeriodStruct::_initSegmentCabin

Definition at line 118 of file [FlightPeriodStruct.hpp](#).

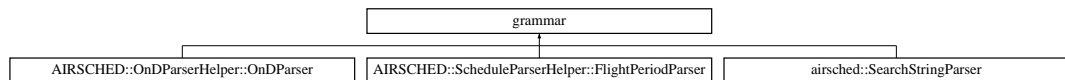
Referenced by [AIRSCHED::ScheduleParserHelper::storeFClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentCabin](#)

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

- [airsched/bom/FlightPeriodStruct.hpp](#)
- [airsched/bom/FlightPeriodStruct.cpp](#)

25.23 grammar Class Reference

Inheritance diagram for grammar:



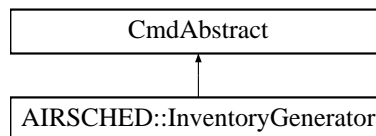
The documentation for this class was generated from the following file:

- [airsched/command/OnDParseHelper.hpp](#)

25.24 AIRSCHED::InventoryGenerator Class Reference

```
#include <airsched/command/InventoryGenerator.hpp>
```

Inheritance diagram for AIRSCHED::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]

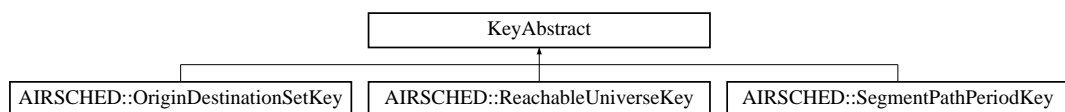
Definition at line 38 of file [InventoryGenerator.hpp](#).

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

- [airsched/command/InventoryGenerator.hpp](#)
- [airsched/command/InventoryGenerator.cpp](#)

25.25 KeyAbstract Class Reference

Inheritance diagram for KeyAbstract:



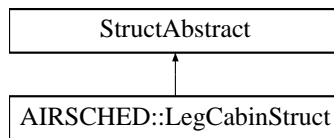
The documentation for this class was generated from the following file:

- [airsched/bom/ReachableUniverseKey.hpp](#)

25.26 AIRSCHED::LegCabinStruct Struct Reference

```
#include <airsched/bom/LegCabinStruct.hpp>
```

Inheritance diagram for AIRSCHED::LegCabinStruct:



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 AIRSCHED::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 AIRSCHED::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 [AIRSCHED::LegStruct::describe\(\)](#).

25.26.3 Member Data Documentation

25.26.3.1 stdair::CabinCode_T AIRSCHED::LegCabinStruct::_cabinCode

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

Referenced by [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#).

25.26.3.2 stdair::CabinCapacity_T AIRSCHED::LegCabinStruct::_capacity

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

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeCapacity::operator>\(\)\(\)](#).

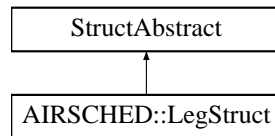
The documentation for this struct was generated from the following files:

- [airsched/bom/LegCabinStruct.hpp](#)
- [airsched/bom/LegCabinStruct.cpp](#)

25.27 AIRSCHED::LegStruct Struct Reference

```
#include <airsched/bom/LegStruct.hpp>
```

Inheritance diagram for AIRSCHED::LegStruct:



Public Member Functions

- void [fill](#) (const stdair::Date_T &iRefDate, stdair::LegDate &) const
- const std::string [describe](#) () const
- [LegStruct](#) ()

Public Attributes

- 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 AIRSCHED::LegStruct::LegStruct ()

Default Constructor.

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

25.27.3 Member Function Documentation

25.27.3.1 void AIRSCHED::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 [_boardingDateOffset](#), [_boardingTime](#), [_elapsed](#), [_offDateOffset](#), [_offPoint](#), and [_offTime](#).

25.27.3.2 const std::string AIRSCHED::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 [AIRSCHED::LegCabinStruct::describe\(\)](#).

Referenced by [AIRSCHED::FlightPeriodStruct::describe\(\)](#).

25.27.4 Member Data Documentation

25.27.4.1 stdair::AirportCode.T AIRSCHED::LegStruct::_boardingPoint

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

Referenced by [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegB](#)

25.27.4.2 stdair::DateOffset.T AIRSCHED::LegStruct::_boardingDateOffset

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

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#).

25.27.4.3 stdair::Duration.T AIRSCHED::LegStruct::_boardingTime

Definition at line 28 of file [LegStruct.hpp](#).

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#).

25.27.4.4 stdair::AirportCode_T AIRSCHED::LegStruct::_offPoint

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

Referenced by [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::store](#).

25.27.4.5 stdair::DateOffset_T AIRSCHED::LegStruct::_offDateOffset

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

Referenced by [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::store](#).

25.27.4.6 stdair::Duration_T AIRSCHED::LegStruct::_offTime

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

Referenced by [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::store](#).

25.27.4.7 stdair::Duration_T AIRSCHED::LegStruct::_elapsed

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

Referenced by [describe\(\)](#), [fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#).

25.27.4.8 LegCabinStructList_T AIRSCHED::LegStruct::_cabinList

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

Referenced by [describe\(\)](#), [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegBoa](#).

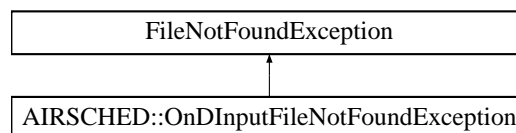
The documentation for this struct was generated from the following files:

- [airsched/bom/LegStruct.hpp](#)
- [airsched/bom/LegStruct.cpp](#)

25.28 AIRSCHED::OnInputFileNotFoundException Class Reference

```
#include <airsched/AIRSCHED_Types.hpp>
```

Inheritance diagram for AIRSCHED::OnInputFileNotFoundException:



Public Member Functions

- [OnInputFileNotFoundException](#) (const std::string &iWhat)

25.28.1 Detailed Description

The O&D input file cannot be retrieved.

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

25.28.2 Constructor & Destructor Documentation

25.28.2.1 AIRSCHED::OnDInputFileNotFoundException::OnDInputFileNotFoundException (
const std::string & *iWhat*) [inline]

Constructor.

Definition at line 40 of file [AIRSCHED_Types.hpp](#).

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

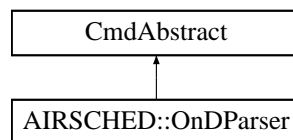
- [airsched/AIRSCHED_Types.hpp](#)

25.29 AIRSCHED::OnDParser Class Reference

Class wrapping the parser entry point.

```
#include <airsched/command/OnDParser.hpp>
```

Inheritance diagram for AIRSCHED::OnDParser:



Static Public Member Functions

- static void [generateOnDPeriods](#) (const stdair::Filename_T &, stdair::BomRoot &)

25.29.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 23 of file [OnDParser.hpp](#).

25.29.2 Member Function Documentation

25.29.2.1 void AIRSCHED::OnDParser::generateOnDPeriods (const stdair::Filename_T &
iFilename, stdair::BomRoot & *ioBomRoot*) [static]

Parse the CSV file describing the O&D.

Parameters

<i>const</i>	std::string& The file-name of the CSV-formatted fare input file and the container.
--------------	--

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

References [AIRSCHED::OnDPeriodFileParser::generateOnDPeriods\(\)](#).

Referenced by [AIRSCHED::AIRSCHED_Service::parseAndLoad\(\)](#).

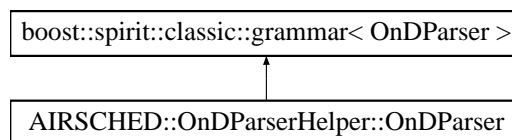
The documentation for this class was generated from the following files:

- [airsched/command/OnDParser.hpp](#)
- [airsched/command/OnDParser.cpp](#)

25.30 AIRSCHED::OnDParserHelper::OnDParser Struct Reference

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

Inheritance diagram for AIRSCHED::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; Last-Date); 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 AIRSCHED::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& AIRSCHED::OnDParserHelper::OnDParser::_bomRoot

Definition at line 145 of file [OnDParserHelper.hpp](#).

25.30.3.2 OnDPeriodStruct& AIRSCHED::OnDParserHelper::OnDParser::_onDPeriod

Definition at line 146 of file [OnDParserHelper.hpp](#).

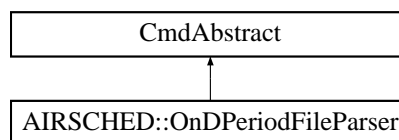
The documentation for this struct was generated from the following files:

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

25.31 AIRSCHED::OnDPeriodFileParser Class Reference

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

Inheritance diagram for AIRSCHED::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 AIRSCHED::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 AIRSCHED::OnDPeriodFileParser::generateOnDPeriods ()

Parse the input file and generate the O&D-Periods.

Definition at line 378 of file [OnDParserHelper.cpp](#).

Referenced by [AIRSCHED::OnDParser::generateOnDPeriods\(\)](#).

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

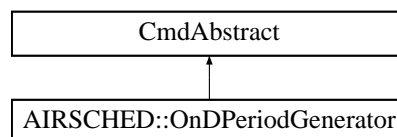
- [airsched/command/OnDParserHelper.hpp](#)
- [airsched/command/OnDParserHelper.cpp](#)

25.32 AIRSCHED::OnDPeriodGenerator Class Reference

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

```
#include <airsched/command/OnDPeriodGenerator.hpp>
```

Inheritance diagram for AIRSCHED::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](#).

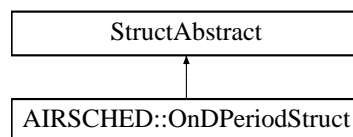
The documentation for this class was generated from the following files:

- [airsched/command/OnDPeriodGenerator.hpp](#)
- [airsched/command/OnDPeriodGenerator.cpp](#)

25.33 AIRSCHED::OnDPeriodStruct Struct Reference

```
#include <airsched/bom/OnDPeriodStruct.hpp>
```

Inheritance diagram for AIRSCHED::OnDPeriodStruct:



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 15 of file [OnDPeriodStruct.hpp](#).

25.33.2 Constructor & Destructor Documentation

25.33.2.1 AIRSCHED::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 & AIRSCHED::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 AIRSCHED::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 [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#).

25.33.3.3 stdair::Duration_T AIRSCHED::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 [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#).

25.33.3.4 const std::string AIRSCHED::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 AIRSCHED::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 AIRSCHED::OnDPeriodStruct::_origin

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

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

25.33.4.2 stdair::AirportCode_T AIRSCHED::OnDPeriodStruct::_destination

Definition at line 42 of file [OnDPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), and [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#).

25.33.4.3 stdair::DatePeriod_T AIRSCHED::OnDPeriodStruct::_datePeriod

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

Referenced by [describe\(\)](#), and [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#).

25.33.4.4 stdair::Duration_T AIRSCHED::OnDPeriodStruct::_timeRangeStart

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

Referenced by [describe\(\)](#), and [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#).

25.33.4.5 stdair::Duration_T AIRSCHED::OnDPeriodStruct::_timeRangeEnd

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

Referenced by [describe\(\)](#), and [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#).

25.33.4.6 stdair::NbOfAirlines_T AIRSCHED::OnDPeriodStruct::_nbOfAirlines

Definition at line 46 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), and [AIRSCHED::OnDParserHelper::store](#)

25.33.4.7 stdair::AirlineCode_T AIRSCHED::OnDPeriodStruct::_airlineCode

Definition at line 47 of file [OnDPeriodStruct.hpp](#).

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

25.33.4.8 stdair::ClassCode_T AIRSCHED::OnDPeriodStruct::_classCode

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

Referenced by [describe\(\)](#), [describeTSKey\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

25.33.4.9 stdair::AirlineCodeList_T AIRSCHED::OnDPeriodStruct::_airlineCodeList

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

Referenced by [getFirstAirlineCode\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

25.33.4.10 stdair::ClassCodeList_T AIRSCHED::OnDPeriodStruct::_classCodeList

Definition at line 50 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), and [AIRSCHED::OnDParserHelper::store](#)

25.33.4.11 stdair::Date_T AIRSCHED::OnDPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 53 of file [OnDPeriodStruct.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#).

25.33.4.12 stdair::Date_T AIRSCHED::OnDPeriodStruct::_dateRangeEnd

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

Referenced by [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#).

25.33.4.13 unsigned int AIRSCHED::OnDPeriodStruct::_itYear

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

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

25.33.4.14 unsigned int AIRSCHED::OnDPeriodStruct::_itMonth

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

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

25.33.4.15 unsigned int AIRSCHED::OnDPeriodStruct::_itDay

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

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

25.33.4.16 long AIRSCHED::OnDPeriodStruct::_itHours

Staging Time.

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

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

25.33.4.17 long AIRSCHED::OnDPeriodStruct::_itMinutes

Definition at line 61 of file [OnDPeriodStruct.hpp](#).

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

25.33.4.18 long AIRSCHED::OnDPeriodStruct::_itSeconds

Definition at line 62 of file [OnDPeriodStruct.hpp](#).

Referenced by [getTime\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#) and [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#).

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

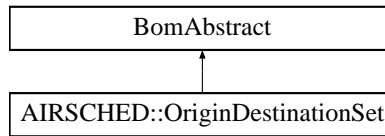
- [airsched/bom/OnDPeriodStruct.hpp](#)
- [airsched/bom/OnDPeriodStruct.cpp](#)

25.34 AIRSCHED::OriginDestinationSet Class Reference

Class representing a simple sub-network.

```
#include <airsched/bom/OriginDestinationSet.hpp>
```

Inheritance diagram for AIRSCHED::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`
`AIRSCHED::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 `AIRSCHED::OriginDestinationSet::OriginDestinationSet (const Key_T & iKey)`
`[protected]`

Main constructor.

Definition at line 31 of file [OriginDestinationSet.cpp](#).

25.34.3.2 `AIRSCHED::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& AIRSCHED::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 std::air::AirportCode_T& AIRSCHED::OriginDestinationSet::getDestination ()`
`const` `[inline]`

Get the destination airport (i.e., the primary key).

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

References [_key](#), and [AIRSCHED::OriginDestinationSetKey::getOffPoint\(\)](#).

25.34.4.3 `stdair::BomAbstract* const AIRSCHED::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& AIRSCHED::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 AIRSCHED::OriginDestinationSet::toStream (std::ostream & ioOut) const`
[inline]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 98 of file [OriginDestinationSet.hpp](#).

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

25.34.4.6 `void AIRSCHED::OriginDestinationSet::fromStream (std::istream & ioIn)`
[inline]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Definition at line 107 of file [OriginDestinationSet.hpp](#).

25.34.4.7 `std::string AIRSCHED::OriginDestinationSet::toString () const`

Get the serialised version of the Business Object.

Definition at line 40 of file [OriginDestinationSet.cpp](#).

References [_key](#), and [AIRSCHED::OriginDestinationSetKey::toString\(\)](#).

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

25.34.4.8 `const std::string AIRSCHED::OriginDestinationSet::describeKey () const`
`[inline]`

Get a string describing the key.

Definition at line 118 of file [OriginDestinationSet.hpp](#).

References [_key](#), and [AIRSCHED::OriginDestinationSetKey::toString\(\)](#).

25.34.4.9 `template<class Archive > void AIRSCHED::OriginDestinationSet::serialize (Archive`
`& ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 59 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 AIRSCHED::OriginDestinationSet::_key` `[protected]`

Primary key (destination airport code).

Definition at line 167 of file [OriginDestinationSet.hpp](#).

Referenced by [describeKey\(\)](#), [getDestination\(\)](#), [getKey\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

25.34.6.2 `stdair::BomAbstract* AIRSCHED::OriginDestinationSet::_parent`
`[protected]`

Pointer on the parent ([ReachableUniverse](#)) object.

Definition at line 172 of file [OriginDestinationSet.hpp](#).

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

25.34.6.3 stdair::HolderMap_T AIRSCHED::OriginDestinationSet::_holderMap [protected]

Map holding the children ([SegmentPathPeriod](#) objects).

Definition at line 177 of file [OriginDestinationSet.hpp](#).

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

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

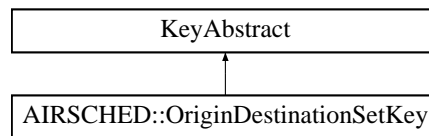
- [airsched/bom/OriginDestinationSet.hpp](#)
- [airsched/bom/OriginDestinationSet.cpp](#)

25.35 AIRSCHED::OriginDestinationSetKey Struct Reference

Structure representing the key of a sub-network.

```
#include <airsched/bom/OriginDestinationSetKey.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::OriginDestinationSetKey::OriginDestinationSetKey (const stdair::AirportCode_T & iDestination)

Constructor.

Definition at line 26 of file [OriginDestinationSetKey.cpp](#).

25.35.2.2 AIRSCHED::OriginDestinationSetKey::OriginDestinationSetKey (const OriginDestinationSetKey & iKey)

Copy constructor.

Definition at line 32 of file [OriginDestinationSetKey.cpp](#).

25.35.2.3 AIRSCHED::OriginDestinationSetKey::~OriginDestinationSetKey ()

Destructor.

Definition at line 37 of file [OriginDestinationSetKey.cpp](#).

25.35.3 Member Function Documentation

25.35.3.1 const stdair::AirportCode_T& AIRSCHED::OriginDestinationSetKey::getOffPoint () const [inline]

Get the destination airport.

Definition at line 62 of file [OriginDestinationSetKey.hpp](#).

Referenced by [AIRSCHED::OriginDestinationSet::getDestination\(\)](#).

25.35.3.2 void AIRSCHED::OriginDestinationSetKey::toStream (std::ostream & ioOut) const

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Definition at line 41 of file [OriginDestinationSetKey.cpp](#).

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

25.35.3.3 void AIRSCHED::OriginDestinationSetKey::fromStream (std::istream & ioIn)

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Definition at line 46 of file [OriginDestinationSetKey.cpp](#).

25.35.3.4 `const std::string AIRSCHED::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 [AIRSCHED::OriginDestinationSet::describeKey\(\)](#), [toStream\(\)](#), and [AIRSCHED::OriginDestinationSet::toS](#)

25.35.3.5 `template<class Archive > void AIRSCHED::OriginDestinationSetKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 69 of file [OriginDestinationSetKey.cpp](#).

25.35.4 Friends And Related Function Documentation

25.35.4.1 `friend class boost::serialization::access [friend]`

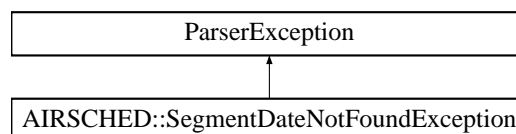
Definition at line 31 of file [OriginDestinationSetKey.hpp](#).

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

- [airsched/bom/OriginDestinationSetKey.hpp](#)
- [airsched/bom/OriginDestinationSetKey.cpp](#)

25.36 ParserException Class Reference

Inheritance diagram for ParserException:



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

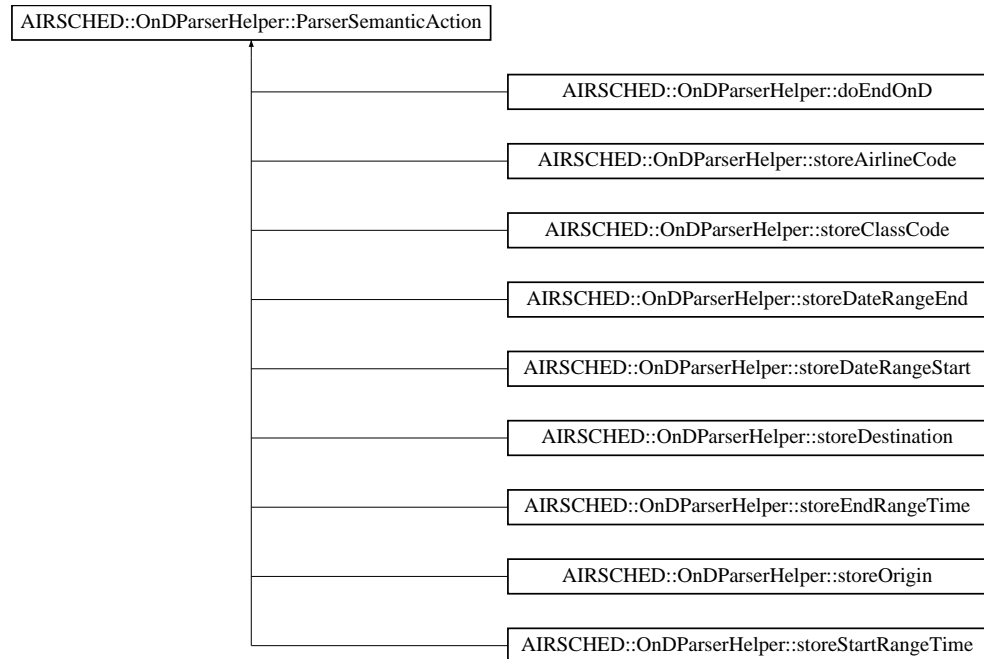
- [airsched/AIRSCHED_Types.hpp](#)

25.37 AIRSCHED::OnDParserHelper::ParserSemanticAction Struct Reference

25.37 AIRSCHED::OnDParserHelper::ParserSemanticAction Struct Reference

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

Inheritance diagram for AIRSCHED::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 AIRSCHED::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& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#) and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

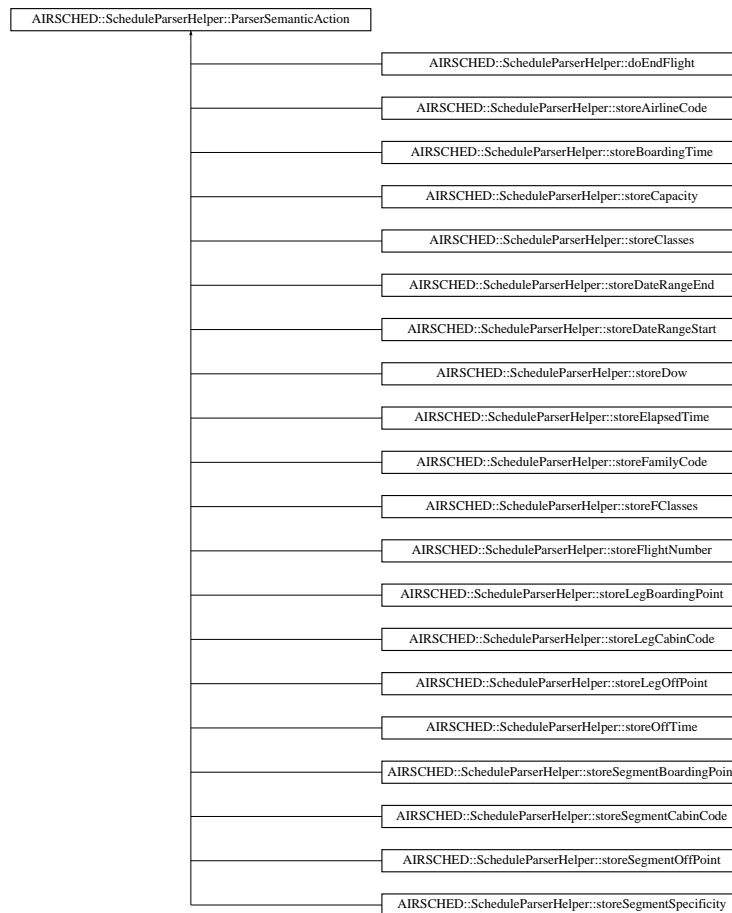
The documentation for this struct was generated from the following files:

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

25.38 AIRSCHED::ScheduleParserHelper::ParserSemanticAction Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 30 of file [ScheduleParserHelper.hpp](#).

25.38.2 Constructor & Destructor Documentation

25.38.2.1 AIRSCHED::ScheduleParserHelper::ParserSemanticAction::ParserSemanticAction (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 23 of file [ScheduleParserHelper.cpp](#).

25.38.3 Member Data Documentation

25.38.3.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOnTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirline::operator\(\)](#).

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

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

25.39 airsched::Passenger_T Struct Reference

```
#include <airsched/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 airsched::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 airsched::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 airsched::Passenger_T::display() const [inline]

Definition at line 80 of file [BookingRequestParser.hpp](#).

References [_labels](#), [_number](#), and [_type](#).

Referenced by [airsched::SearchString_T::display\(\)](#).

25.39.5 Member Data Documentation

25.39.5.1 const std::string airsched::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` `airsched::Passenger_T::_type`

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

Referenced by [display\(\)](#), [airsched::store_pet_passenger_type::operator\(\)\(\)](#), [airsched::store_child_passenger_type::operator\(\)\(\)](#), and [airsched::store_adult_passenger_type::operator\(\)\(\)](#).

25.39.5.3 `unsigned short` `airsched::Passenger_T::_number`

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

Referenced by [display\(\)](#), and [airsched::store_passenger_number::operator\(\)\(\)](#).

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

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

25.40 `airsched::Place_T` Struct Reference

```
#include <airsched/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 `airsched::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 airsched::Place_T::display () const [inline]

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

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

Referenced by [airsched::SearchString_T::display\(\)](#).

25.40.4 Member Data Documentation

25.40.4.1 std::string airsched::Place_T::_name

Definition at line 13 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), and [airsched::store_place_element::operator\(\)\(\)](#).

25.40.4.2 std::string airsched::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:

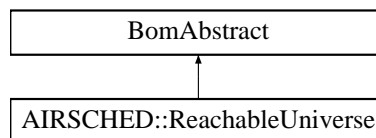
- [airsched/batches/BookingRequestParser.hpp](#)

25.41 AIRSCHED::ReachableUniverse Class Reference

Class representing the root of the schedule-related BOM tree.

```
#include <airsched/bom/ReachableUniverse.hpp>
```

Inheritance diagram for AIRSCHED::ReachableUniverse:



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 AIRSCHED::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 AIRSCHED::ReachableUniverse::ReachableUniverse (const Key_T & iKey) [protected]

Main constructor.

Definition at line 32 of file [ReachableUniverse.cpp](#).

25.41.3.2 AIRSCHED::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& AIRSCHED::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& AIRSCHED::ReachableUniverse::getOrigin () const [inline]

Get the (origin) airport (i.e., the primary key).

Definition at line 70 of file [ReachableUniverse.hpp](#).

References [_key](#), and [AIRSCHED::ReachableUniverseKey::getBoardingPoint\(\)](#).

25.41.4.3 stdair::BomAbstract* const AIRSCHED::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& AIRSCHED::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&
AIRSCHED::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 AIRSCHED::ReachableUniverse::toStream (std::ostream & ioOut) const
[inline]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 103 of file [ReachableUniverse.hpp](#).

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

25.41.4.7 `void AIRSCHED::ReachableUniverse::fromStream (std::istream & ioIn)
[inline]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Definition at line 112 of file [ReachableUniverse.hpp](#).

25.41.4.8 `std::string AIRSCHED::ReachableUniverse::toString () const`

Get the serialised version of the Business Object.

Definition at line 41 of file [ReachableUniverse.cpp](#).

References [_key](#), and [AIRSCHED::ReachableUniverseKey::toString\(\)](#).

Referenced by [AIRSCHED::BomDisplay::csvDisplay\(\)](#), and [toStream\(\)](#).

25.41.4.9 `const std::string AIRSCHED::ReachableUniverse::describeKey () const
[inline]`

Get a string describing the key.

Definition at line 123 of file [ReachableUniverse.hpp](#).

References [_key](#), and [AIRSCHED::ReachableUniverseKey::toString\(\)](#).

25.41.4.10 `template<class Archive > void AIRSCHED::ReachableUniverse::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 60 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 AIRSCHED::ReachableUniverse::_key` [`protected`]

Primary key (origin airport code).

Definition at line 173 of file [ReachableUniverse.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [getOrigin\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

25.41.6.2 `stdair::BomAbstract* AIRSCHED::ReachableUniverse::_parent` [`protected`]

Pointer on the parent (BomRoot) object.

Definition at line 178 of file [ReachableUniverse.hpp](#).

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

25.41.6.3 `stdair::HolderMap_T AIRSCHED::ReachableUniverse::_holderMap` [`protected`]

Map holding the children ([OriginDestinationSet](#) objects).

Definition at line 183 of file [ReachableUniverse.hpp](#).

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

25.41.6.4 SegmentPathPeriodListList_T AIRSCHED::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 190 of file [ReachableUniverse.hpp](#).

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

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

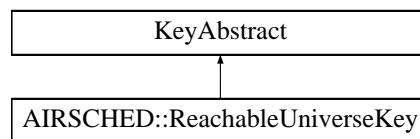
- [airsched/bom/ReachableUniverse.hpp](#)
- [airsched/bom/ReachableUniverse.cpp](#)

25.42 AIRSCHED::ReachableUniverseKey Struct Reference

Structure representing the key of the schedule-related BOM tree root.

```
#include <airsched/bom/ReachableUniverseKey.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::ReachableUniverseKey::ReachableUniverseKey (const `stdair::AirportCode.T` & *iOrigin*)

Constructor.

Definition at line 32 of file [ReachableUniverseKey.cpp](#).

25.42.2.2 AIRSCHED::ReachableUniverseKey::ReachableUniverseKey (const `ReachableUniverseKey` & *iKey*)

Copy constructor.

Definition at line 26 of file [ReachableUniverseKey.cpp](#).

25.42.2.3 AIRSCHED::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` & AIRSCHED::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 [AIRSCHED::ReachableUniverse::getOrigin\(\)](#).

25.42.3.2 void AIRSCHED::ReachableUniverseKey::toStream (`std::ostream` & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Definition at line 41 of file [ReachableUniverseKey.cpp](#).

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

25.42.3.3 void AIRSCHED::ReachableUniverseKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Definition at line 46 of file [ReachableUniverseKey.cpp](#).

25.42.3.4 const std::string AIRSCHED::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 [AIRSCHED::ReachableUniverse::describeKey\(\)](#), [toStream\(\)](#), and [AIRSCHED::ReachableUniverse::toString\(\)](#).

25.42.3.5 template<class Archive > void AIRSCHED::ReachableUniverseKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 69 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](#).

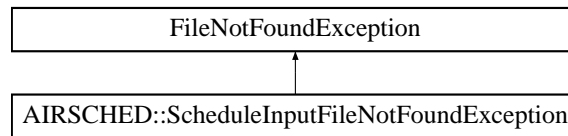
The documentation for this struct was generated from the following files:

- [airsched/bom/ReachableUniverseKey.hpp](#)
- [airsched/bom/ReachableUniverseKey.cpp](#)

25.43 AIRSCHED::ScheduleInputFileNotFound Exception Class Reference

```
#include <airsched/AIRSCHED_Types.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleInputFileNotFound Exception:



Public Member Functions

- [ScheduleInputFileNotFound Exception](#) (const std::string &iWhat)

25.43.1 Detailed Description

The schedule input file cannot be retrieved.

Definition at line 47 of file [AIRSCHED_Types.hpp](#).

25.43.2 Constructor & Destructor Documentation

25.43.2.1 `AIRSCHED::ScheduleInputFileNotFound Exception::ScheduleInputFileNotFound Exception (const std::string & iWhat) [inline]`

Constructor.

Definition at line 53 of file [AIRSCHED_Types.hpp](#).

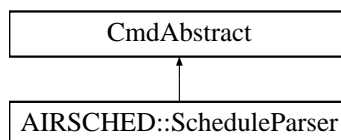
The documentation for this class was generated from the following file:

- [airsched/AIRSCHED_Types.hpp](#)

25.44 AIRSCHED::ScheduleParser Class Reference

```
#include <airsched/command/ScheduleParser.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParser:



Static Public Member Functions

- static void [generateInventories](#) (const stdair::Filename_T &, stdair::BomRoot &)

25.44.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 21 of file [ScheduleParser.hpp](#).

25.44.2 Member Function Documentation

25.44.2.1 `void AIRSCHED::ScheduleParser::generateInventories (const stdair::Filename_T & iFilename, stdair::BomRoot & ioBomRoot) [static]`

Parse the CSV file describing the airline schedules for the simulator, and generates the inventories accordingly.

Parameters

<code>const</code>	<code>stdair::Filename_T&</code> The file-name of the CSV-formatted schedule input file.
	Root of the BOM tree.
<code>stdair::BomRoot</code>	

Definition at line 18 of file [ScheduleParser.cpp](#).

References [AIRSCHED::SegmentPathGenerator::createSegmentPathNetwork\(\)](#), and [AIRSCHED::FlightPeriodFileParser::parseAndLoad\(\)](#).

Referenced by [AIRSCHED::AIRSCHED_Service::parseAndLoad\(\)](#).

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

- [airsched/command/ScheduleParser.hpp](#)
- [airsched/command/ScheduleParser.cpp](#)

25.45 `airsched::SearchString_T` Struct Reference

```
#include <airsched/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 airsched::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 airsched::SearchString_T::display () const [inline]

Definition at line 105 of file [BookingRequestParser.hpp](#).

References [_airlineList](#), [_dateList](#), [_passengerList](#), [_placeList](#), [_tmpPlace](#), [airsched::Passenger_T::display\(\)](#), [airsched::Airline_T::display\(\)](#), [airsched::Date_T::display\(\)](#), and [airsched::Place_T::display\(\)](#).

25.45.4 Member Data Documentation

25.45.4.1 PlaceList_T airsched::SearchString_T::_placeList

Definition at line 96 of file [BookingRequestParser.hpp](#).

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

25.45.4.2 DateList_T airsched::SearchString_T::_dateList

Definition at line 97 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), and [airsched::store_date::operator\(\)\(\)](#).

25.45.4.3 AirlineList_T airsched::SearchString_T::_airlineList

Definition at line 98 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), [airsched::store_airline_name::operator\(\)\(\)](#), and [airsched::store_airline_code::operator\(\)\(\)](#).

25.45.4.4 PassengerList_T airsched::SearchString_T::_passengerList

Definition at line 99 of file [BookingRequestParser.hpp](#).

Referenced by [display\(\)](#), [airsched::store_pet_passenger_type::operator\(\)\(\)](#), [airsched::store_child_passenger_type::operator\(\)\(\)](#), and [airsched::store_adult_passenger_type::operator\(\)\(\)](#).

25.45.4.5 Place_T airsched::SearchString_T::_tmpPlace

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

Referenced by [display\(\)](#), and [airsched::store_place_element::operator\(\)](#).

25.45.4.6 Date_T airsched::SearchString_T::_tmpDate

Definition at line 138 of file [BookingRequestParser.hpp](#).

Referenced by [airsched::store_date::operator\(\)](#).

25.45.4.7 Airline_T airsched::SearchString_T::_tmpAirline

Definition at line 139 of file [BookingRequestParser.hpp](#).

Referenced by [airsched::store_airline_name::operator\(\)](#), [airsched::store_airline_code::operator\(\)](#), and [airsched::store_airline_sign::operator\(\)](#).

25.45.4.8 Passenger_T airsched::SearchString_T::_tmpPassenger

Definition at line 140 of file [BookingRequestParser.hpp](#).

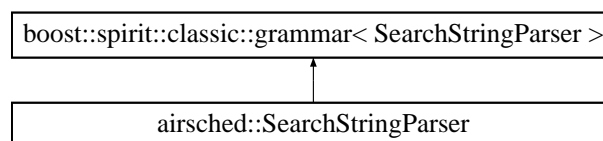
Referenced by [airsched::store_pet_passenger_type::operator\(\)](#), [airsched::store_child_passenger_type::operator\(\)](#), [airsched::store_adult_passenger_type::operator\(\)](#), and [airsched::store_passenger_number::operator\(\)](#).

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

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

25.46 airsched::SearchStringParser Struct Reference

Inheritance diagram for airsched::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 `airsched::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](#) & `airsched::SearchStringParser::_searchString`

Definition at line 369 of file [BookingRequestParser.cpp](#).

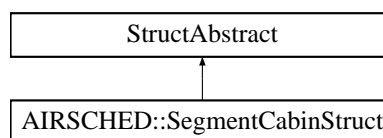
The documentation for this struct was generated from the following file:

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

25.47 AIRSCHED::SegmentCabinStruct Struct Reference

```
#include <airsched/bom/SegmentCabinStruct.hpp>
```

Inheritance diagram for AIRSCHED::SegmentCabinStruct:



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](#)
- [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 AIRSCHED::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 AIRSCHED::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 [AIRSCHED::SegmentStruct::describe\(\)](#).

25.47.3 Member Data Documentation

25.47.3.1 `stdair::CabinCode_T AIRSCHED::SegmentCabinStruct::_cabinCode`

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

Referenced by [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#), [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#).

25.47.3.2 `stdair::ClassList_String_T AIRSCHED::SegmentCabinStruct::_classes`

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

Referenced by [describe\(\)](#), [AIRSCHED::SegmentPeriodHelper::fill\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#).

25.47.3.3 `stdair::FamilyCode.T AIRSCHED::SegmentCabinStruct::_itFamilyCode`

Definition at line 28 of file [SegmentCabinStruct.hpp](#).

Referenced by [AIRSCHED::ScheduleParserHelper::storeFClasses::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#).

25.47.3.4 FareFamilyStructList_T AIRSCHED::SegmentCabinStruct::_fareFamilies

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

Referenced by [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#).

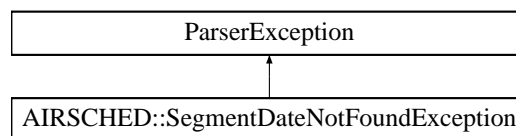
The documentation for this struct was generated from the following files:

- [airsched/bom/SegmentCabinStruct.hpp](#)
- [airsched/bom/SegmentCabinStruct.cpp](#)

25.48 AIRSCHED::SegmentDateNotFoundException Class Reference

```
#include <airsched/AIRSCHED_Types.hpp>
```

Inheritance diagram for AIRSCHED::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 [AIRSCHED_Types.hpp](#).

25.48.2 Constructor & Destructor Documentation

25.48.2.1 AIRSCHED::SegmentDateNotFoundException::SegmentDateNotFoundException (const std::string & iWhat) [inline]

Constructor.

Definition at line 28 of file [AIRSCHED_Types.hpp](#).

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

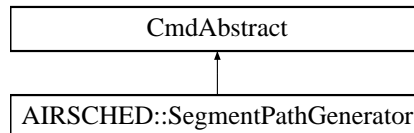
- [airsched/AIRSCHED_Types.hpp](#)

25.49 AIRSCHED::SegmentPathGenerator Class Reference

Class handling the generation / instantiation of the network BOM.

```
#include <airsched/command/SegmentPathGenerator.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::SegmentPathGenerator::createSegmentPathNetwork (const stdair::BomRoot & *iBomRoot*) [static]

Generate the segment path network.

Definition at line 26 of file [SegmentPathGenerator.cpp](#).

Referenced by [AIRSCHED::AIRSCHED_Service::buildSampleBom\(\)](#), and [AIRSCHED::ScheduleParser::generateInvent](#)

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

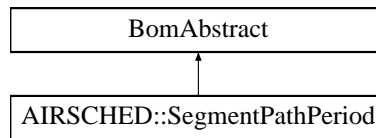
- [airsched/command/SegmentPathGenerator.hpp](#)
- [airsched/command/SegmentPathGenerator.cpp](#)

25.50 AIRSCHED::SegmentPathPeriod Class Reference

Class representing a segment/path.

```
#include <airsched/bom/SegmentPathPeriod.hpp>
```

Inheritance diagram for AIRSCHED::SegmentPathPeriod:



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 AIRSCHED::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 AIRSCHED::SegmentPathPeriod::SegmentPathPeriod (const Key_T & iKey) [protected]

Main constructor.

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

25.50.3.2 AIRSCHED::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& AIRSCHED::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 AIRSCHED::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& AIRSCHED::SegmentPathPeriod::getDeparturePeriod ()`
`const [inline]`

Get the departure period (part of the primary key).

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

References [_key](#), and [AIRSCHED::SegmentPathPeriodKey::getPeriod\(\)](#).

Referenced by [connectWithAnotherSegment\(\)](#), and [isDepartureDateValid\(\)](#).

25.50.4.4 `const DateOffsetList_T& AIRSCHED::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 [AIRSCHED::SegmentPathPeriodKey::getBoardingDateOffsetList\(\)](#).

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

25.50.4.5 `const stdair::NbOfSegments_T AIRSCHED::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 [AIRSCHED::SegmentPathPeriodKey::getNbOfSegments\(\)](#).

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

25.50.4.6 `const stdair::NbOfAirlines_T& AIRSCHED::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 [AIRSCHED::SegmentPathPeriodKey::getNbOfAirlines\(\)](#).

25.50.4.7 `const stdair::Duration_T& AIRSCHED::SegmentPathPeriod::getElapsedTime () const`
`[inline]`

Get the elapsed time (part of the primary key).

Definition at line 92 of file [SegmentPathPeriod.hpp](#).

References [_key](#), and [AIRSCHED::SegmentPathPeriodKey::getElapsedTime\(\)](#).

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

25.50.4.8 `const stdair::Duration_T& AIRSCHED::SegmentPathPeriod::getBoardingTime ()
const [inline]`

Get the boarding time (part of the primary key).

Definition at line 97 of file [SegmentPathPeriod.hpp](#).

References [_key](#), and [AIRSCHED::SegmentPathPeriodKey::getBoardingTime\(\)](#).

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

25.50.4.9 `const stdair::HolderMap_T& AIRSCHED::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 * AIRSCHED::SegmentPathPeriod::getLastSegmentPeriod ()
const`

Get the last SegmentPeriod object of the list.

Return a NULL pointer if the list is empty.

Definition at line 77 of file [SegmentPathPeriod.cpp](#).

Referenced by [connectWithAnotherSegment\(\)](#), and [getDestination\(\)](#).

25.50.4.11 `stdair::SegmentPeriod * AIRSCHED::SegmentPathPeriod::getFirstSegmentPeriod () const`

Get the first SegmentPeriod object of the list.

Return a NULL pointer if the list is empty.

Definition at line 95 of file [SegmentPathPeriod.cpp](#).

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

25.50.4.12 `const stdair::AirportCode_T & AIRSCHED::SegmentPathPeriod::getDestination ()
const`

Get the destination of the segment path (i.e., the destination of the last segment).

Definition at line 113 of file [SegmentPathPeriod.cpp](#).

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

25.50.4.13 `SegmentPathPeriodKey AIRSCHED::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 149 of file [SegmentPathPeriod.cpp](#).

References [checkCircle\(\)](#), [getBoardingDateOffsetList\(\)](#), [getBoardingTime\(\)](#), [getDeparturePeriod\(\)](#), [getElapsedTime\(\)](#), [getFirstSegmentPeriod\(\)](#), [getLastSegmentPeriod\(\)](#), [getNbOfSegments\(\)](#), [AIRSCHED::SegmentPathPeriodKey::setBoardingDateOffsetList\(\)](#), [AIRSCHED::SegmentPathPeriodKey::setElapsedTime\(\)](#), and [AIRSCHED::SegmentPathPeriodKey::setPeriod\(\)](#).

25.50.4.14 `bool AIRSCHED::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.

Definition at line 275 of file [SegmentPathPeriod.cpp](#).

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

25.50.4.15 `bool AIRSCHED::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 121 of file [SegmentPathPeriod.cpp](#).

25.50.4.16 `bool AIRSCHED::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 294 of file [SegmentPathPeriod.cpp](#).

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

25.50.4.17 `void AIRSCHED::SegmentPathPeriod::toStream (std::ostream & ioOut) const [inline]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Definition at line 176 of file [SegmentPathPeriod.hpp](#).

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

25.50.4.18 void AIRSCHED::SegmentPathPeriod::fromStream (std::istream & *ioIn*)
[inline]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Definition at line 185 of file [SegmentPathPeriod.hpp](#).

25.50.4.19 std::string AIRSCHED::SegmentPathPeriod::toString () const

Get the serialised version of the Business Object.

Definition at line 52 of file [SegmentPathPeriod.cpp](#).

References [_key](#), and [AIRSCHED::SegmentPathPeriodKey::toString\(\)](#).

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

25.50.4.20 const std::string AIRSCHED::SegmentPathPeriod::describeKey () const
[inline]

Get a string describing the key.

Definition at line 196 of file [SegmentPathPeriod.hpp](#).

References [_key](#), and [AIRSCHED::SegmentPathPeriodKey::toString\(\)](#).

25.50.4.21 template<class Archive > void AIRSCHED::SegmentPathPeriod::serialize (Archive
& *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 71 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 AIRSCHED::SegmentPathPeriod::_key [protected]

Primary key (segment/path characteristics: scheduled period, number of segments, number of airlines, elapsed time, boarding time).

Definition at line 248 of file [SegmentPathPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingDateOffsetList\(\)](#), [getBoardingTime\(\)](#), [getDeparturePeriod\(\)](#), [getElapsedTime\(\)](#), [getKey\(\)](#), [getNbOfAirlines\(\)](#), [getNbOfSegments\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

25.50.6.2 stdair::BomAbstract* AIRSCHED::SegmentPathPeriod::_parent [protected]

Pointer on the parent ([OriginDestinationSet](#)) object.

Definition at line 253 of file [SegmentPathPeriod.hpp](#).

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

25.50.6.3 stdair::HolderMap_T AIRSCHED::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 261 of file [SegmentPathPeriod.hpp](#).

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

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

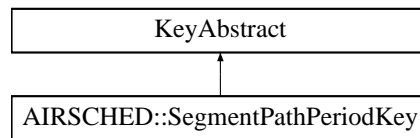
- [airsched/bom/SegmentPathPeriod.hpp](#)
- [airsched/bom/SegmentPathPeriod.cpp](#)

25.51 AIRSCHED::SegmentPathPeriodKey Struct Reference

Structure representing the key of a segment/path.

```
#include <airsched/bom/SegmentPathPeriodKey.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::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 39 of file [SegmentPathPeriodKey.cpp](#).

25.51.2.2 AIRSCHED::SegmentPathPeriodKey::SegmentPathPeriodKey ()

Default constructor.

Definition at line 21 of file [SegmentPathPeriodKey.cpp](#).

25.51.2.3 AIRSCHED::SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey & *iSPPK*)

Copy constructor.

Definition at line 29 of file [SegmentPathPeriodKey.cpp](#).

25.51.2.4 AIRSCHED::SegmentPathPeriodKey::~~SegmentPathPeriodKey ()

Destructor.

Definition at line 52 of file [SegmentPathPeriodKey.cpp](#).

25.51.3 Member Function Documentation

25.51.3.1 const stdair::PeriodStruct& AIRSCHED::SegmentPathPeriodKey::getPeriod () const
[inline]

Get the active days-of-week.

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

Referenced by [AIRSCHED::SegmentPathPeriod::getDeparturePeriod\(\)](#).

25.51.3.2 const DateOffsetList_T& AIRSCHED::SegmentPathPeriodKey::getBoardingDateOffsetList () const [inline]

Get the list of boarding date off-sets.

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

Referenced by [AIRSCHED::SegmentPathPeriod::getBoardingDateOffsetList\(\)](#).

25.51.3.3 const stdair::NbOfSegments_T AIRSCHED::SegmentPathPeriodKey::getNbOfSegments () const [inline]

Get the number of segments.

Definition at line 82 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::getNbOfSegments\(\)](#).

```
25.51.3.4  const stdair::NbOfAirlines_T& AIRSCHED::SegmentPathPeriodKey::getNbOfAirlines (
           ) const  [inline]
```

Get the number of airlines.

Definition at line 89 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::getNbOfAirlines\(\)](#).

```
25.51.3.5  const stdair::Duration_T& AIRSCHED::SegmentPathPeriodKey::getElapsedTime ( )
           const  [inline]
```

Get the elapsed time.

Definition at line 96 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::getElapsedTime\(\)](#).

```
25.51.3.6  const stdair::Duration_T& AIRSCHED::SegmentPathPeriodKey::getBoardingTime ( )
           const  [inline]
```

Get the boarding time.

Definition at line 103 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::getBoardingTime\(\)](#).

```
25.51.3.7  void AIRSCHED::SegmentPathPeriodKey::setPeriod ( const stdair::PeriodStruct &
           iPeriod )  [inline]
```

Set the active days-of-week.

Definition at line 111 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

```
25.51.3.8  void AIRSCHED::SegmentPathPeriodKey::setBoardingDateOffsetList ( const
           DateOffsetList_T & iList )  [inline]
```

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

Referenced by [AIRSCHED::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

```
25.51.3.9  void AIRSCHED::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 AIRSCHED::SegmentPathPeriodKey::setElapsedTime (const stdair::Duration_T & *iElapsed*) [inline]

Set the elapsed time.

Definition at line 125 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

25.51.3.11 void AIRSCHED::SegmentPathPeriodKey::setBoardingTime (const stdair::Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 130 of file [SegmentPathPeriodKey.hpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::connectWithAnotherSegment\(\)](#).

25.51.3.12 const bool AIRSCHED::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 AIRSCHED::SegmentPathPeriodKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Definition at line 56 of file [SegmentPathPeriodKey.cpp](#).

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

25.51.3.14 void AIRSCHED::SegmentPathPeriodKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Definition at line 61 of file [SegmentPathPeriodKey.cpp](#).

25.51.3.15 const std::string AIRSCHED::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 65 of file [SegmentPathPeriodKey.cpp](#).

Referenced by [AIRSCHED::SegmentPathPeriod::describeKey\(\)](#), [toStream\(\)](#), and [AIRSCHED::SegmentPathPeriod::toSt](#)

25.51.3.16 `template<class Archive > void AIRSCHED::SegmentPathPeriodKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 94 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:

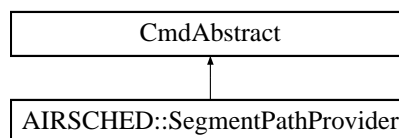
- [airsched/bom/SegmentPathPeriodKey.hpp](#)
- [airsched/bom/SegmentPathPeriodKey.cpp](#)

25.52 AIRSCHED::SegmentPathProvider Class Reference

Class building the travel solutions from airline schedules.

```
#include <airsched/command/SegmentPathProvider.hpp>
```

Inheritance diagram for AIRSCHED::SegmentPathProvider:



Friends

- class [AIRSCHED_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 AIRSCHED_Service [friend]

Definition at line 28 of file [SegmentPathProvider.hpp](#).

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

- [airsched/command/SegmentPathProvider.hpp](#)
- [airsched/command/SegmentPathProvider.cpp](#)

25.53 AIRSCHED::SegmentPeriodHelper Class Reference

```
#include <airsched/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 19 of file [SegmentPeriodHelper.hpp](#).

25.53.2 Member Function Documentation

25.53.2.1 void AIRSCHED::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 [AIRSCHED::SegmentCabinStruct::_cabinCode](#), [AIRSCHED::SegmentStruct::_cabinList](#), and [AIRSCHED::SegmentCabinStruct::_classes](#).

25.53.2.2 void AIRSCHED::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 [AIRSCHED::LegStruct::_boardingPoint](#), [AIRSCHED::LegStruct::_offDateOffset](#), [AIRSCHED::LegStruct::_offPoint](#), and [AIRSCHED::LegStruct::_offTime](#).

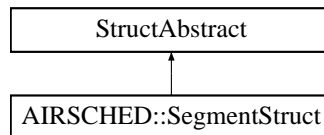
The documentation for this class was generated from the following files:

- [airsched/bom/SegmentPeriodHelper.hpp](#)
- [airsched/bom/SegmentPeriodHelper.cpp](#)

25.54 AIRSCHED::SegmentStruct Struct Reference

```
#include <airsched/bom/SegmentStruct.hpp>
```

Inheritance diagram for AIRSCHED::SegmentStruct:



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 AIRSCHED::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 AIRSCHED::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 [AIRSCHED::SegmentCabinStruct::describe\(\)](#).

Referenced by [AIRSCHED::FlightPeriodStruct::describe\(\)](#).

25.54.3 Member Data Documentation

25.54.3.1 `stdair::AirportCode.T AIRSCHED::SegmentStruct::_boardingPoint`

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

Referenced by [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#), [AIRSCHED::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRSCHED::FlightPeriodStruct::buildSegments\(\)](#), [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentBo](#)

25.54.3.2 `stdair::Date.T AIRSCHED::SegmentStruct::_boardingDate`

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

25.54.3.3 `stdair::Duration.T AIRSCHED::SegmentStruct::_boardingTime`

Definition at line 28 of file [SegmentStruct.hpp](#).

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

25.54.3.4 `stdair::AirportCode.T AIRSCHED::SegmentStruct::_offPoint`

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

Referenced by [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#), [AIRSCHED::FlightPeriodStruct::addSegmentCabin\(\)](#), [AIRSCHED::FlightPeriodStruct::buildSegments\(\)](#), [describe\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeSegmentOf](#)

25.54.3.5 `stdair::Date.T AIRSCHED::SegmentStruct::_offDate`

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

25.54.3.6 `stdair::Duration.T AIRSCHED::SegmentStruct::_offTime`

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

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

25.54.3.7 `stdair::Duration.T AIRSCHED::SegmentStruct::_elapsed`

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

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

25.54.3.8 `SegmentCabinStructList.T AIRSCHED::SegmentStruct::_cabinList`

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

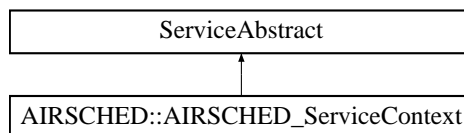
Referenced by [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#), [AIRSCHED::FlightPeriodStruct::addSegmentCabin\(\)](#), [describe\(\)](#), and [AIRSCHED::SegmentPeriodHelper::fill\(\)](#).

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

- [airsched/bom/SegmentStruct.hpp](#)
- [airsched/bom/SegmentStruct.cpp](#)

25.55 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



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

- [airsched/service/AIRSCHED_ServiceContext.hpp](#)

25.56 AIRSCHED::ServiceAbstract Class Reference

```
#include <airsched/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 AIRSCHED::ServiceAbstract::~~ServiceAbstract () [inline, virtual]

Destructor.

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

25.56.2.2 AIRSCHED::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 AIRSCHED::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 22 of file [ServiceAbstract.hpp](#).

Referenced by [operator<<\(\)](#).

25.56.3.2 virtual void AIRSCHED::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

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

Referenced by [operator>>\(\)](#).

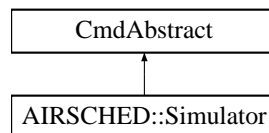
The documentation for this class was generated from the following file:

- [airsched/service/ServiceAbstract.hpp](#)

25.57 AIRSCHED::Simulator Class Reference

```
#include <airsched/command/Simulator.hpp>
```

Inheritance diagram for AIRSCHED::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 AIRSCHED::Simulator::simulate (stdair::BomRoot & *ioBomRoot*)
[static]

Perform a small simulation, which uses the Airline Schedule.

Parameters

<i>stdair::BomRoot</i>	Root of the BOM tree.
------------------------	-----------------------

Definition at line 19 of file [Simulator.cpp](#).

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

- airsched/command/[Simulator.hpp](#)
- airsched/command/[Simulator.cpp](#)

25.58 airsched::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 `airsched::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 airsched::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 [airsched::SearchString_T::_passengerList](#), [_searchString](#), [airsched::SearchString_T::_tmpPassenger](#), [airsched::Passenger_T::_type](#), and [airsched::Passenger_T::ADULT](#).

25.58.4 Member Data Documentation

25.58.4.1 `SearchString_T& airsched::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:

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

25.59 airsched::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 `airsched::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 airsched::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 [airsched::SearchString_T::_airlineList](#), [airsched::Airline_T::_code](#), [_searchString](#), and [airsched::SearchString_T::_tmpAirline](#).

25.59.4 Member Data Documentation

25.59.4.1 `SearchString_T& airsched::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:

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

25.60 airsched::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 airsched::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 airsched::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 [airsched::SearchString_T::_airlineList](#), [airsched::Airline_T::_name](#), [_searchString](#), and [airsched::SearchString_T::_tmpAirline](#).

25.60.4 Member Data Documentation

25.60.4.1 SearchString_T& airsched::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:

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

25.61 airsched::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 `airsched::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 airsched::store_airline_sign::operator() (bool iAirlineSign) const [inline]`

Parse the airline sign.

Definition at line 83 of file [BookingRequestParser.cpp](#).

References [airsched::Airline_T::_isPreferred](#), [_searchString](#), and [airsched::SearchString_T::_tmpAirline](#).

25.61.4 Member Data Documentation

25.61.4.1 `SearchString_T& airsched::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:

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

25.62 airsched::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 `airsched::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 airsched::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 [airsched::SearchString_T::_passengerList](#), [_searchString](#), [airsched::SearchString_T::_tmpPassenger](#), [airsched::Passenger_T::_type](#), and [airsched::Passenger_T::CHILD](#).

25.62.4 Member Data Documentation

25.62.4.1 `SearchString_T& airsched::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:

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

25.63 airsched::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 `airsched::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 airschd::store_date::operator() (iterator_t iStr, iterator_t iStrEnd) const`
`[inline]`

Parse the date.

Definition at line 64 of file [BookingRequestParser.cpp](#).

References [airsched::Date_T::_date](#), [airsched::SearchString_T::_dateList](#), [_searchString](#), [airsched::SearchString_T::_tmpDate](#), and [airsched::Date_T::getDate\(\)](#).

25.63.4 Member Data Documentation

25.63.4.1 `SearchString_T & airschd::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:

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

25.64 `airsched::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 `airsched::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 aircshed::store_passenger_number::operator() (unsigned int iNumber) const` `[inline]`

Parse number of passengers.

Definition at line 136 of file [BookingRequestParser.cpp](#).

References [airsched::Passenger_T::_number](#), [_searchString](#), and [airsched::SearchString_T::_tmpPassenger](#).

25.64.4 Member Data Documentation

25.64.4.1 `SearchString_T & aircshed::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:

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

25.65 `airsched::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 `airsched::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 airsched::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 [airsched::SearchString_T::_passengerList](#), [_searchString](#), [airsched::SearchString_T::_tmpPassenger](#), [airsched::Passenger_T::_type](#), and [airsched::Passenger_T::PET](#).

25.65.4 Member Data Documentation

25.65.4.1 `SearchString_T& airsched::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:

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

25.66 airsched::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](#).

25.67 AIRSCHED::OnDParserHelper::storeAirlineCode Struct Reference 258

25.66.2 Constructor & Destructor Documentation

25.66.2.1 `airsched::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 airschd::store_place_element::operator() (iterator_t iStr, iterator_t iStrEnd) const [inline]`

Parse the place.

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

References [airsched::Place_T::_name](#), [_searchString](#), and [airsched::SearchString_T::_tmpPlace](#).

25.66.4 Member Data Documentation

25.66.4.1 `SearchString_T& airschd::store_place_element::_searchString`

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

Referenced by [operator>\(\)](#).

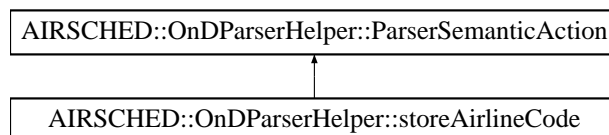
The documentation for this struct was generated from the following file:

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

25.67 AIRSCHED::OnDParserHelper::storeAirlineCode Struct Reference

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

Inheritance diagram for AIRSCHED::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 AIRSCHED::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 AIRSCHED::OnDParserHelper::storeAirlineCode::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 144 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_airlineCode](#), [AIRSCHED::OnDPeriodStruct::_airlineCodeList](#), [AIRSCHED::OnDPeriodStruct::_nbOfAirlines](#), and [AIRSCHED::OnDParserHelper::ParserSemanticAction::onDPeriod](#).

25.67.4 Member Data Documentation

25.67.4.1 [OnDPeriodStruct](#)& [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#) [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

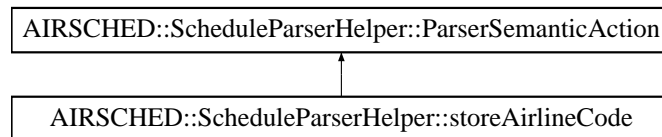
- [airsched/command/OnDParserHelper.hpp](#)
- [airsched/command/OnDParserHelper.cpp](#)

25.68 AIRSCHED::ScheduleParserHelper::storeAirlineCode Struct Reference 260

25.68 AIRSCHED::ScheduleParserHelper::storeAirlineCode Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 38 of file [ScheduleParserHelper.hpp](#).

25.68.2 Constructor & Destructor Documentation

25.68.2.1 AIRSCHED::ScheduleParserHelper::storeAirlineCode::storeAirlineCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 29 of file [ScheduleParserHelper.cpp](#).

25.68.3 Member Function Documentation

25.68.3.1 void AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 34 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_airlineCode](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::flightPeriod](#), and [AIRSCHED::FlightPeriodStruct::_legList](#).

25.69 AIRSCHED::ScheduleParserHelper::storeBoardingTime Struct Reference

25.68.4 Member Data Documentation

25.68.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOnTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentOnTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegSegmentOnTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [operator\(\)](#).

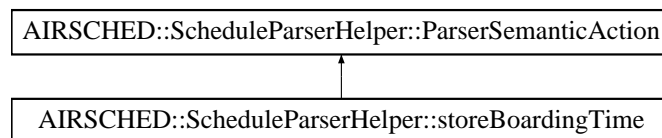
The documentation for this struct was generated from the following files:

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

25.69 AIRSCHED::ScheduleParserHelper::storeBoardingTime Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeBoardingTime:



Public Member Functions

- [storeBoardingTime](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.69 AIRSCHED::ScheduleParserHelper::storeBoardingTime Struct Reference 262

25.69.1 Detailed Description

Store the boarding time.

Definition at line 94 of file [ScheduleParserHelper.hpp](#).

25.69.2 Constructor & Destructor Documentation

25.69.2.1 AIRSCHED::ScheduleParserHelper::storeBoardingTime::storeBoardingTime (FlightPeriodStruct & ioFlightPeriod)

Actor Constructor.

Definition at line 152 of file [ScheduleParserHelper.cpp](#).

25.69.3 Member Function Documentation

25.69.3.1 void AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 157 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::LegStruct::_boardingTime](#), [AIRSCHED::FlightPeriodStruct::_dateOffset](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itLeg](#), [AIRSCHED::FlightPeriodStruct::_itSeconds](#), and [AIRSCHED::FlightPeriodStruct::getTime\(\)](#).

25.69.4 Member Data Documentation

25.69.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_ flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlight::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

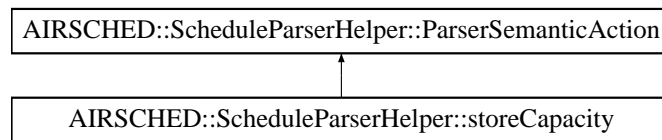
- [airsched/command/ScheduleParserHelper.hpp](#)

- [airsched/command/ScheduleParserHelper.cpp](#)

25.70 AIRSCHED::ScheduleParserHelper::storeCapacity Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 126 of file [ScheduleParserHelper.hpp](#).

25.70.2 Constructor & Destructor Documentation

25.70.2.1 AIRSCHED::ScheduleParserHelper::storeCapacity::storeCapacity ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

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

25.70.3 Member Function Documentation

25.70.3.1 void AIRSCHED::ScheduleParserHelper::storeCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 229 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::LegStruct::_cabinList](#), [AIRSCHED::LegCabinStruct::_capacity](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itLeg](#), and [AIRSCHED::FlightPeriodStruct::_itLegCabin](#).

25.70.4 Member Data Documentation

25.70.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightPeriod::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

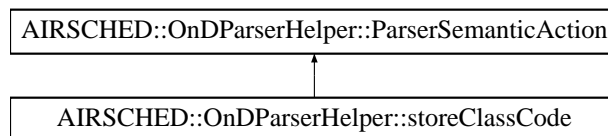
The documentation for this struct was generated from the following files:

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

25.71 AIRSCHED::OnDParserHelper::storeClassCode Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeClassCode:



Public Member Functions

- [storeClassCode](#) ([OnDPeriodStruct](#) &)
- [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 AIRSCHED::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 AIRSCHED::OnDParserHelper::storeClassCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 177 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_classCode](#), [AIRSCHED::OnDPeriodStruct::_classCodeList](#), and [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#).

25.71.4 Member Data Documentation

25.71.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

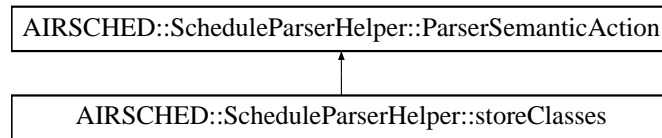
The documentation for this struct was generated from the following files:

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

25.72 AIRSCHED::ScheduleParserHelper::storeClasses Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 169 of file [ScheduleParserHelper.hpp](#).

25.72.2 Constructor & Destructor Documentation

25.72.2.1 AIRSCHED::ScheduleParserHelper::storeClasses::storeClasses ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 306 of file [ScheduleParserHelper.cpp](#).

25.72.3 Member Function Documentation

25.72.3.1 void AIRSCHED::ScheduleParserHelper::storeClasses::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 311 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRSCHED::SegmentCabinStruct::_classes](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itSegment](#), [AIRSCHED::FlightPeriodStruct::_itSegmentCabin](#), and [AIRSCHED::FlightPeriodStruct::addSegmentCabin\(\)](#)

25.72.4 Member Data Documentation

25.72.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_-
flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElaps::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#) and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

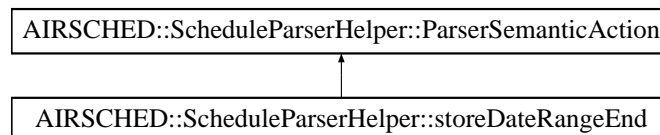
The documentation for this struct was generated from the following files:

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

25.73 AIRSCHED::ScheduleParserHelper::storeDateRangeEnd Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 62 of file [ScheduleParserHelper.hpp](#).

25.73.2 Constructor & Destructor Documentation

Actor Constructor.

Definition at line 72 of file [ScheduleParserHelper.cpp](#).

25.73.3 Member Function Documentation

Actor Function (functor).

Definition at line 77 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_dateRange](#), [AIRSCHED::FlightPeriodStruct::_dateRangeEnd](#), [AIRSCHED::FlightPeriodStruct::_dateRangeStart](#), [AIRSCHED::ScheduleParserHelper::ParserSemantic::flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itSeconds](#), and [AIRSCHED::FlightPeriodStruct::getDate\(\)](#).

25.73.4 Member Data Documentation

Actor Context.

Definition at line 34 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHEd::ScheduleParserHelper::doEndFlight::operator(), AIRSCHEd::ScheduleParserHelper::store
AIRSCHEd::ScheduleParserHelper::storeFamilyCode::operator(), AIRSCHEd::ScheduleParserHelper::storeClasses:
AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRSCHEd::ScheduleParserHelper::storeS
AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint::operator(), AIRSCHEd::ScheduleParserHelper::sto
AIRSCHEd::ScheduleParserHelper::storeCapacity::operator(), AIRSCHEd::ScheduleParserHelper::storeLegCabinCo
AIRSCHEd::ScheduleParserHelper::storeElapsedTime::operator(), AIRSCHEd::ScheduleParserHelper::storeOffTime
AIRSCHEd::ScheduleParserHelper::storeBoardingTime::operator(), AIRSCHEd::ScheduleParserHelper::storeLegOff
AIRSCHEd::ScheduleParserHelper::storeLegBoardingPoint::operator(), AIRSCHEd::ScheduleParserHelper::storeDo
operator(), AIRSCHEd::ScheduleParserHelper::storeDateRangeStart::operator(), AIRSCHEd::ScheduleParserHelp
and AIRSCHEd::ScheduleParserHelper::storeAirlineCode::operator()).

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

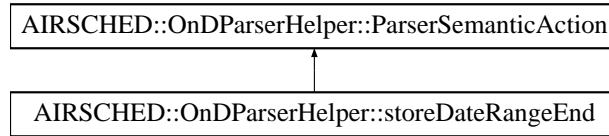
- `airsched/command/ScheduleParserHelper.hpp`
- `airsched/command/ScheduleParserHelper.cpp`

25.74 AIRSCHED::OnDParserHelper::storeDateRangeEnd Struct Reference

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

25.74 AIRSCHED::OnDParserHelper::storeDateRangeEnd Struct Reference 269

Inheritance diagram for AIRSCHED::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 AIRSCHED::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 AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator() ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Actor Function (functor).

Definition at line 88 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_datePeriod](#), [AIRSCHED::OnDPeriodStruct::_dateRangeEnd](#), [AIRSCHED::OnDPeriodStruct::_dateRangeStart](#), [AIRSCHED::OnDPeriodStruct::_itSeconds](#), [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#), and [AIRSCHED::OnDPeriodStruct::get](#)

25.74.4 Member Data Documentation

25.74.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

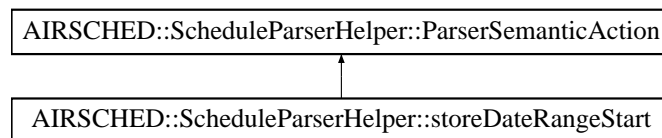
The documentation for this struct was generated from the following files:

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

25.75 AIRSCHED::ScheduleParserHelper::storeDateRangeStart Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeDateRangeStart:



Public Member Functions

- [storeDateRangeStart](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.75.1 Detailed Description

Store the start of the date range.

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

25.76 AIRSCHED::OnDParserHelper::storeDateRangeStart Struct Reference 271

25.75.2 Constructor & Destructor Documentation

25.75.2.1 AIRSCHED::ScheduleParserHelper::storeDateRangeStart::storeDateRangeStart (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 57 of file [ScheduleParserHelper.cpp](#).

25.75.3 Member Function Documentation

25.75.3.1 void AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 62 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_dateRangeStart](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itSeconds](#), and [AIRSCHED::FlightPeriodStruct::getDate\(\)](#).

25.75.4 Member Data Documentation

25.75.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_ flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

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

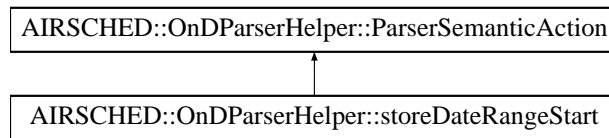
- [airsched/command/ScheduleParserHelper.hpp](#)
- [airsched/command/ScheduleParserHelper.cpp](#)

25.76 AIRSCHED::OnDParserHelper::storeDateRangeStart Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeDateRangeStart:

25.76 AIRSCHED::OnDParserHelper::storeDateRangeStart Struct Reference 272



Public Member Functions

- [storeDateRangeStart](#) ([OnDPeriodStruct](#) &)
- void [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [OnDPeriodStruct](#) & [_onDPeriod](#)

25.76.1 Detailed Description

Store the start of the date range.

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

25.76.2 Constructor & Destructor Documentation

25.76.2.1 AIRSCHED::OnDParserHelper::storeDateRangeStart::storeDateRangeStart ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

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

25.76.3 Member Function Documentation

25.76.3.1 void AIRSCHED::OnDParserHelper::storeDateRangeStart::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 71 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_dateRangeStart](#), [AIRSCHED::OnDPeriodStruct::_itSeconds](#), [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#), and [AIRSCHED::OnDPeriodStruct::get](#)

25.76.4 Member Data Documentation

25.76.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOnDPeriod::operator\(\)](#).

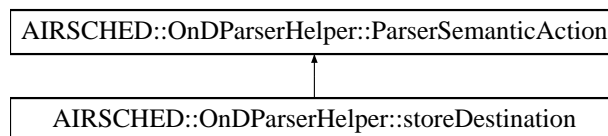
The documentation for this struct was generated from the following files:

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

25.77 AIRSCHED::OnDParserHelper::storeDestination Struct Reference

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

Inheritance diagram for AIRSCHED::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 AIRSCHED::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 AIRSCHED::OnDParserHelper::storeDestination::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 55 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_destination](#), and [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#).

25.77.4 Member Data Documentation

25.77.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [operator\(\)\(\)](#), and [AIRSCHED::OnDParserHelper::storeDow::operator\(\)\(\)](#).

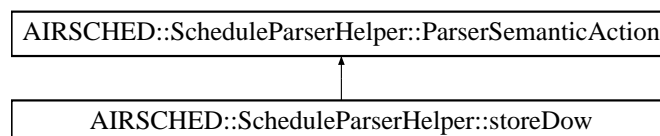
The documentation for this struct was generated from the following files:

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

25.78 AIRSCHED::ScheduleParserHelper::storeDow Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeDow:



25.79 AIRSCHED::ScheduleParserHelper::storeElapsedTime Struct Reference 276

[AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeLegCabinCo](#)
[AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeOffTime](#)
[AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeLegOff](#)
[AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#)(), [operator\(\)](#)(), [AIRSCHED::ScheduleParserHel](#)
[AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeFligh](#)
and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#)).

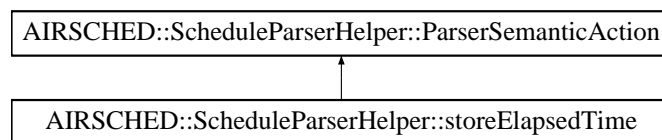
The documentation for this struct was generated from the following files:

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

25.79 AIRSCHED::ScheduleParserHelper::storeElapsedTime Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::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 110 of file [ScheduleParserHelper.hpp](#).

25.79.2 Constructor & Destructor Documentation

25.79.2.1 AIRSCHED::ScheduleParserHelper::storeElapsedTime::storeElapsedTime ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 191 of file [ScheduleParserHelper.cpp](#).

25.79.3 Member Function Documentation

Actor Function (functor).

Definition at line 196 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_dateOffset](#), [AIRSCHED::LegStruct::_elapsed](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itLeg](#), [AIRSCHED::FlightPeriodStruct::_itSeconds](#), [AIRSCHED::LegStruct::_offDateOffset](#), and [AIRSCHED::FlightPeriodStruct::getTime\(\)](#).

25.79.4 Member Data Documentation

25.79.4.1 **FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod** [inherited]

Actor Context.

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

Referenced by AIRSCHEd::ScheduleParserHelper::doEndFlight::operator(), AIRSCHEd::ScheduleParserHelper::store
AIRSCHEd::ScheduleParserHelper::storeFamilyCode::operator(), AIRSCHEd::ScheduleParserHelper::storeClasses:
AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRSCHEd::ScheduleParserHelper::storeS
AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint::operator(), AIRSCHEd::ScheduleParserHelper::sto
AIRSCHEd::ScheduleParserHelper::storeCapacity::operator(), AIRSCHEd::ScheduleParserHelper::storeLegCabinCo
operator(), AIRSCHEd::ScheduleParserHelper::storeOffTime::operator(), AIRSCHEd::ScheduleParserHelper::storel
AIRSCHEd::ScheduleParserHelper::storeLegOffPoint::operator(), AIRSCHEd::ScheduleParserHelper::storeLegBoar
AIRSCHEd::ScheduleParserHelper::storeDow::operator(), AIRSCHEd::ScheduleParserHelper::storeDateRangeEnd:
AIRSCHEd::ScheduleParserHelper::storeDateRangeStart::operator(), AIRSCHEd::ScheduleParserHelper::storeFligh
and AIRSCHEd::ScheduleParserHelper::storeAirlineCode::operator()).

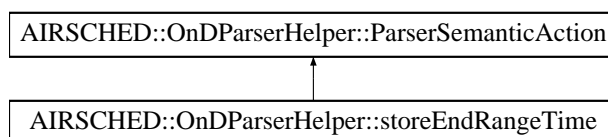
The documentation for this struct was generated from the following files:

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

25.80 AIRSCHED::OnDParserHelper::storeEndRangeTime Struct Reference

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

Inheritance diagram for AIRSCHED::OnDPParserHelper::storeEndRangeTime:



25.80 AIRSCHED::OnDParserHelper::storeEndRangeTime Struct Reference 278

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 AIRSCHED::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 AIRSCHED::OnDParserHelper::storeEndRangeTime::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 129 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_itSeconds](#), [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#), [AIRSCHED::OnDPeriodStruct::_timeRangeEnd](#), and [AIRSCHED::OnDPeriodStruct::getTime\(\)](#).

25.80.4 Member Data Documentation

25.80.4.1 [OnDPeriodStruct](#)& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#)(), [operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeStartRange::operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#)(), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#)(), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)](#).

25.81 AIRSCHED::ScheduleParserHelper::storeFamilyCode Struct Reference 279

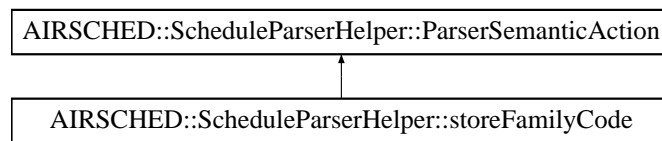
The documentation for this struct was generated from the following files:

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

25.81 AIRSCHED::ScheduleParserHelper::storeFamilyCode Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFamilyCode:



Public Member Functions

- [storeFamilyCode](#) ([FlightPeriodStruct](#) &)
- [operator\(\)](#) (int iCode) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.81.1 Detailed Description

Store the parsed family code.

Definition at line 177 of file [ScheduleParserHelper.hpp](#).

25.81.2 Constructor & Destructor Documentation

25.81.2.1 AIRSCHED::ScheduleParserHelper::storeFamilyCode::storeFamilyCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 331 of file [ScheduleParserHelper.cpp](#).

25.81.3 Member Function Documentation

25.81.3.1 void AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

25.82 AIRSCHED::ScheduleParserHelper::storeFClasses Struct Reference 280

Definition at line 336 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::SegmentCabinStruct::itFamilyCode](#), and [AIRSCHED::FlightPeriodStruct::_itSegmentCabin](#).

25.81.4 Member Data Documentation

25.81.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightPeriod::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

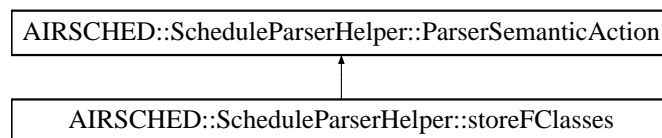
The documentation for this struct was generated from the following files:

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

25.82 AIRSCHED::ScheduleParserHelper::storeFClasses Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFClasses:



Public Member Functions

- [storeFClasses](#) ([FlightPeriodStruct](#) &)
- [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

25.82 AIRSCHED::ScheduleParserHelper::storeFClasses Struct Reference 281

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.82.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 185 of file [ScheduleParserHelper.hpp](#).

25.82.2 Constructor & Destructor Documentation

25.82.2.1 AIRSCHED::ScheduleParserHelper::storeFClasses::storeFClasses ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 344 of file [ScheduleParserHelper.cpp](#).

25.82.3 Member Function Documentation

25.82.3.1 void AIRSCHED::ScheduleParserHelper::storeFClasses::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 349 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRSCHED::ScheduleParserHelper::ParseFlightPeriod](#), [AIRSCHED::SegmentCabinStruct::_itFamilyCode](#), [AIRSCHED::FlightPeriodStruct::_itSegment](#), [AIRSCHED::FlightPeriodStruct::_itSegmentCabin](#), and [AIRSCHED::FlightPeriodStruct::addFareFamily\(\)](#).

25.82.4 Member Data Documentation

25.82.4.1 [FlightPeriodStruct](#)& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [\[inherited\]](#)

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDo](#)

25.83 AIRSCHED::ScheduleParserHelper::storeFlightNumber Struct Reference 282

[AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirline::operator\(\)](#)

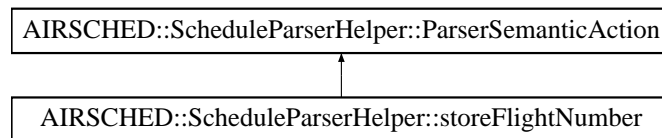
The documentation for this struct was generated from the following files:

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

25.83 AIRSCHED::ScheduleParserHelper::storeFlightNumber Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFlightNumber:



Public Member Functions

- [storeFlightNumber](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (unsigned int iNumber) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.83.1 Detailed Description

Store the parsed flight number.

Definition at line 46 of file [ScheduleParserHelper.hpp](#).

25.83.2 Constructor & Destructor Documentation

25.83.2.1 AIRSCHED::ScheduleParserHelper::storeFlightNumber::storeFlightNumber ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 46 of file [ScheduleParserHelper.cpp](#).

25.84 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

283

25.83.3 Member Function Documentation

25.83.3.1 void AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 51 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_flightNumber](#), and [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::flightPeriod](#).

25.83.4 Member Data Documentation

25.83.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClassesOfService::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeDateRange::operator\(\)\(\)](#).

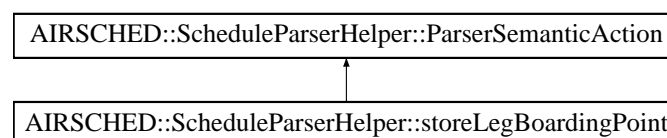
The documentation for this struct was generated from the following files:

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

25.84 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint:



Public Member Functions

- [storeLegBoardingPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.84.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 78 of file [ScheduleParserHelper.hpp](#).

25.84.2 Constructor & Destructor Documentation

25.84.2.1 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::storeLegBoardingPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 107 of file [ScheduleParserHelper.cpp](#).

25.84.3 Member Function Documentation

25.84.3.1 void AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 112 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::LegStruct::_boardingPoint](#), [AIRSCHED::LegStruct::_cabinList](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itLeg](#), [AIRSCHED::FlightPeriodStruct::_legAlreadyDefined](#), [AIRSCHED::FlightPeriodStruct::_legList](#), and [AIRSCHED::FlightPeriodStruct::addAirport\(\)](#).

25.84.4 Member Data Documentation

25.84.4.1 [FlightPeriodStruct](#)& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#).

[AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)\(\)](#).

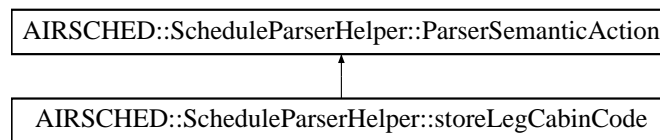
The documentation for this struct was generated from the following files:

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

25.85 AIRSCHED::ScheduleParserHelper::storeLegCabinCode Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegCabinCode:



Public Member Functions

- [storeLegCabinCode](#) ([FlightPeriodStruct](#) &)
- [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.85.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 118 of file [ScheduleParserHelper.hpp](#).

25.85.2 Constructor & Destructor Documentation

25.85.2.1 AIRSCHED::ScheduleParserHelper::storeLegCabinCode::storeLegCabinCode ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

25.86 AIRSCHED::ScheduleParserHelper::storeLegOffPoint Struct Reference²⁸⁶

Definition at line 212 of file [ScheduleParserHelper.cpp](#).

25.85.3 Member Function Documentation

25.85.3.1 void AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator() (char iChar)
const

Actor Function (functor).

Definition at line 217 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::LegCabinStruct::_cabinCode](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRSCHED::FlightPeriodStruct::_itLegCabin](#).

25.85.4 Member Data Documentation

25.85.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightPeriod::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

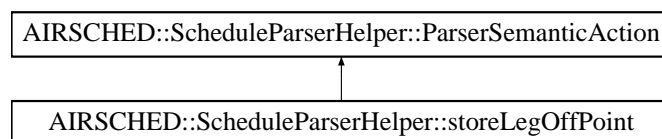
The documentation for this struct was generated from the following files:

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

25.86 AIRSCHED::ScheduleParserHelper::storeLegOffPoint Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegOffPoint:



Public Member Functions

- ## Public Attributes

- ### 25.86.1 Detailed Description

Store the parsed leg off point.

Definition at line 86 of file [ScheduleParserHelper.hpp](#).

25.86.2 Constructor & Destructor Documentation

Actor Constructor.

Definition at line 136 of file [ScheduleParserHelper.cpp](#).

25.86.3 Member Function Documentation

Actor Function (functor).

Definition at line 141 of file ScheduleParserHelper.cpp.

References AIRSCHEd::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHEd::FlightPeriodStruct::itLeg, AIRSCHEd::LegStruct::offPoint, and AIRSCHEd::FlightPeriodStruct::addAirport().

25.86.4 Member Data Documentation

Actor Context.

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

Referenced by AIRSCHEd::ScheduleParserHelper::doEndFlight::operator(), AIRSCHEd::ScheduleParserHelper::store
AIRSCHEd::ScheduleParserHelper::storeFamilyCode::operator(), AIRSCHEd::ScheduleParserHelper::storeClasses:
AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode::operator(), AIRSCHEd::ScheduleParserHelper::storeS
AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint::operator(), AIRSCHEd::ScheduleParserHelper::sto

[AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeLegCabinCo](#)
[AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeOffTime](#)
[AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#)(), [operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::](#)
[AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::](#)
[AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#)(), [AIRSCHED::ScheduleParserHelper::storeFlight](#)
and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#)).

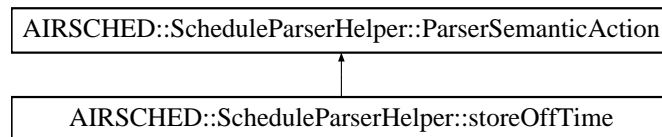
The documentation for this struct was generated from the following files:

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

25.87 AIRSCHED::ScheduleParserHelper::storeOffTime Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeOffTime:



Public Member Functions

- [storeOffTime](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.87.1 Detailed Description

Store the off time.

Definition at line 102 of file [ScheduleParserHelper.hpp](#).

25.87.2 Constructor & Destructor Documentation

25.87.2.1 AIRSCHED::ScheduleParserHelper::storeOffTime::storeOffTime ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 170 of file [ScheduleParserHelper.cpp](#).

25.87.3 Member Function Documentation

25.87.3.1 void AIRSCHED::ScheduleParserHelper::storeOffTime::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 175 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::LegStruct::_boardingDateOffset](#), [AIRSCHED::FlightPeriodStruct::_dateOffset](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::_itLeg](#), [AIRSCHED::FlightPeriodStruct::_itSeconds](#), [AIRSCHED::LegStruct::_offTime](#), and [AIRSCHED::FlightPeriodStruct::getTime\(\)](#).

25.87.4 Member Data Documentation

25.87.4.1 **FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod** [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightPeriod::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

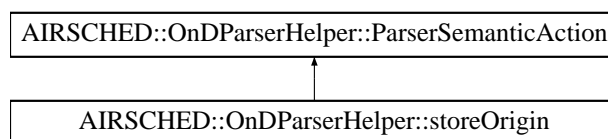
The documentation for this struct was generated from the following files:

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

25.88 AIRSCHED::OnDParserHelper::storeOrigin Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeOrigin:



Public Member Functions

- [storeOrigin](#) ([OnDPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [OnDPeriodStruct](#) & [_onDPeriod](#)

25.88.1 Detailed Description

Store the parsed origin.

Definition at line 42 of file [OnDParserHelper.hpp](#).

25.88.2 Constructor & Destructor Documentation

25.88.2.1 AIRSCHED::OnDParserHelper::storeOrigin::storeOrigin ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

Definition at line 30 of file [OnDParserHelper.cpp](#).

25.88.3 Member Function Documentation

25.88.3.1 void AIRSCHED::OnDParserHelper::storeOrigin::operator() ([iterator_t](#) *iStr*, [iterator_t](#) *iStrEnd*) const

Actor Function (functor).

Definition at line 35 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_airlineCode](#), [AIRSCHED::OnDPeriodStruct::_airlineCodeList](#), [AIRSCHED::OnDPeriodStruct::_classCode](#), [AIRSCHED::OnDPeriodStruct::_classCodeList](#), [AIRSCHED::OnDPeriodStruct::_nbOfAirlines](#), [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#), and [AIRSCHED::OnDPeriodStruct::_origin](#).

25.88.4 Member Data Documentation

25.88.4.1 [OnDPeriodStruct&](#) [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#) [inherited]

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::storeOrigin\(\)](#).

25.89 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference 291

[AIRSCHED::OnDParserHelper::storeStartRangeTime::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)](#) and [operator\(\)](#).

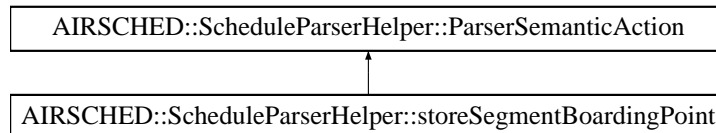
The documentation for this struct was generated from the following files:

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

25.89 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint:



Public Member Functions

- [storeSegmentBoardingPoint](#) ([FlightPeriodStruct](#) &)
- [void operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.89.1 Detailed Description

Store the parsed segment boarding point.

Definition at line [145](#) of file [ScheduleParserHelper.hpp](#).

25.89.2 Constructor & Destructor Documentation

25.89.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line [269](#) of file [ScheduleParserHelper.cpp](#).

25.89.3 Member Function Documentation

25.89.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

Definition at line 274 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::SegmentStruct::_boardingPoint](#), [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), and [AIRSCHED::FlightPeriodStruct::_itSegment](#).

25.89.4 Member Data Documentation

25.89.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirline::operator\(\)\(\)](#).

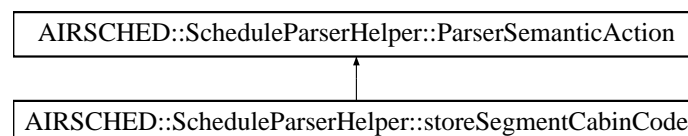
The documentation for this struct was generated from the following files:

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

25.90 AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode:



[AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsed::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDow::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightPeriod::operator\(\)](#) and [AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator\(\)](#).

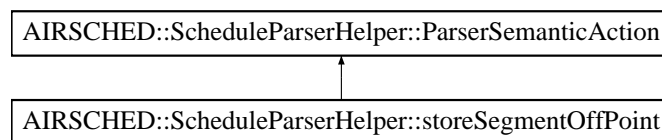
The documentation for this struct was generated from the following files:

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

25.91 AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint:



Public Member Functions

- [storeSegmentOffPoint](#) ([FlightPeriodStruct](#) &)
- [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.91.1 Detailed Description

Store the parsed segment off point.

Definition at line 153 of file [ScheduleParserHelper.hpp](#).

25.91.2 Constructor & Destructor Documentation

25.91.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::storeSegmentOffPoint ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 282 of file [ScheduleParserHelper.cpp](#).

25.91.3 Member Function Documentation

25.91.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator() (iterator_t iStr, iterator_t iStrEnd) const

Actor Function (functor).

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

References [AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod](#), [AIRSCHED::FlightPeriodStruct::itSegment](#), and [AIRSCHED::SegmentStruct::_offPoint](#).

25.91.4 Member Data Documentation

25.91.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClassesAIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)](#), [operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeAIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCoAIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTimeAIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffAIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDoAIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateAIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAir](#)

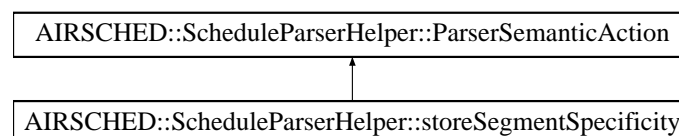
The documentation for this struct was generated from the following files:

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

25.92 AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity Struct Reference

```
#include <airsched/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity:



Public Member Functions

- [storeSegmentSpecificity](#) ([FlightPeriodStruct](#) &)
- void [operator\(\)](#) (char iChar) const

Public Attributes

- [FlightPeriodStruct](#) & [_flightPeriod](#)

25.92.1 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 137 of file [ScheduleParserHelper.hpp](#).

25.92.2 Constructor & Destructor Documentation

25.92.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::storeSegmentSpecificity ([FlightPeriodStruct](#) & *ioFlightPeriod*)

Actor Constructor.

Definition at line 243 of file [ScheduleParserHelper.cpp](#).

25.92.3 Member Function Documentation

25.92.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 248 of file [ScheduleParserHelper.cpp](#).

References [AIRSCHED::FlightPeriodStruct::_airportList](#), [AIRSCHED::FlightPeriodStruct::_airportOrderedList](#), [AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific](#), [AIRSCHED::ScheduleParserHelper::flightPeriod](#), and [AIRSCHED::FlightPeriodStruct::buildSegments\(\)](#).

25.92.4 Member Data Documentation

25.92.4.1 [FlightPeriodStruct](#)& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

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

Referenced by [AIRSCHED::ScheduleParserHelper::doEndFlight::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator\(\)](#), [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeClasses::operator\(\)](#).

25.93 AIRSCHED::OnDParserHelper::storeStartRangeTime Struct Reference 297

[AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeCapacity::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegOffTime::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeDateRange::operator\(\)\(\)](#), [AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator\(\)\(\)](#), and [AIRSCHED::ScheduleParserHelper::storeAirline::operator\(\)\(\)](#)

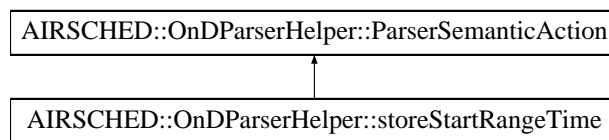
The documentation for this struct was generated from the following files:

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

25.93 AIRSCHED::OnDParserHelper::storeStartRangeTime Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeStartRangeTime:



Public Member Functions

- [storeStartRangeTime](#) ([OnDPeriodStruct](#) &)
- [operator\(\)](#) ([iterator_t](#) iStr, [iterator_t](#) iStrEnd) const

Public Attributes

- [OnDPeriodStruct](#) & [_onDPeriod](#)

25.93.1 Detailed Description

Store the start range time.

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

25.93.2 Constructor & Destructor Documentation

25.93.2.1 AIRSCHED::OnDParserHelper::storeStartRangeTime::storeStartRangeTime ([OnDPeriodStruct](#) & *ioOnDPeriod*)

Actor Constructor.

Definition at line 109 of file [OnDParserHelper.cpp](#).

25.93.3 Member Function Documentation

25.93.3.1 `void AIRSCHED::OnDParserHelper::storeStartRangeTime::operator() (iterator_t iStr, iterator_t iStrEnd) const`

Actor Function (functor).

Definition at line 114 of file [OnDParserHelper.cpp](#).

References [AIRSCHED::OnDPeriodStruct::_itSeconds](#), [AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod](#), [AIRSCHED::OnDPeriodStruct::_timeRangeStart](#), and [AIRSCHED::OnDPeriodStruct::getTime\(\)](#).

25.93.4 Member Data Documentation

25.93.4.1 `OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod` `[inherited]`

Actor Context.

Definition at line 38 of file [OnDParserHelper.hpp](#).

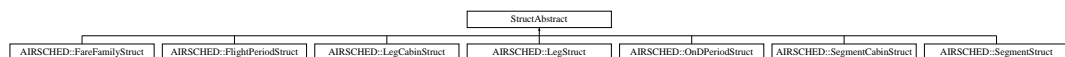
Referenced by [AIRSCHED::OnDParserHelper::doEndOnD::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeClassCode::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeAirlineCode::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [AIRSCHED::OnDParserHelper::storeDestination::operator\(\)\(\)](#), and [AIRSCHED::OnDParserHelper::storeOrigin::operator\(\)\(\)](#).

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

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

25.94 StructAbstract Class Reference

Inheritance diagram for StructAbstract:

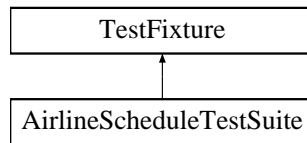


The documentation for this class was generated from the following file:

- [airsched/bom/SegmentStruct.hpp](#)

25.95 TestFixture Class Reference

Inheritance diagram for TestFixture:



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

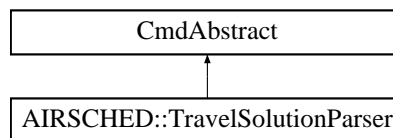
- test/airsched/[AirlineScheduleTestSuite.hpp](#)

25.96 AIRSCHED::TravelSolutionParser Class Reference

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

```
#include <airsched/command/TravelSolutionParser.hpp>
```

Inheritance diagram for AIRSCHED::TravelSolutionParser:



Static Public Member Functions

- static bool [parseInputFileAndBuildBom](#) (const stdair::Filename_T &)

25.96.1 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](#).

25.96.2 Member Function Documentation

25.96.2.1 bool AIRSCHED::TravelSolutionParser::parseInputFileAndBuildBom (const stdair::Filename_T &) [static]

Parse the input values from a CSV-formatted travel solution file.

Parameters

<i>const</i> std::string& iInputFileName	Travel solution file to be parsed.
--	------------------------------------

Returns

bool Whether or not the parsing was successful.

Definition at line 21 of file [TravelSolutionParser.cpp](#).

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

- [airsched/command/TravelSolutionParser.hpp](#)
- [airsched/command/TravelSolutionParser.cpp](#)

26 File Documentation**26.1 airsched/AIRSCHEd_Service.hpp File Reference**

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <airsched/AIRSCHEd_Types.hpp>
```

Classes

- class [AIRSCHEd::AIRSCHEd_Service](#)
Interface for the AirSched Services.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHEd](#)

26.2 AIRSCHEd_Service.hpp

```
00001 #ifndef __AIRSCHEd_SVC_AIRSCHEd_SERVICE_HPP
00002 #define __AIRSCHEd_SVC_AIRSCHEd_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/bom/TravelSolutionTypes.hpp>
00011 // AirSched
00012 #include <airsched/AIRSCHEd_Types.hpp>
00013
00015 namespace stdair {
```

```

00016     class STDAIR_Service;
00017     struct BasLogParams;
00018     struct BasDBParams;
00019     struct BookingRequestStruct;
00020     struct TravelSolutionStruct;
00021 }
00022
00023 namespace AIRSCHED {
00024
00026     class AIRSCHED_ServiceContext;
00027
00028
00032     class AIRSCHED_Service {
00033     public:
00034         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00050         AIRSCHED_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);
00051
00063         AIRSCHED_Service (const stdair::BasLogParams&);
00064
00080         AIRSCHED_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr);
00081
00090         void parseAndLoad (const stdair::Filename_T& iScheduleInputFilename);
00091
00101         void parseAndLoad (const stdair::Filename_T& iScheduleFilename,
00102                             const stdair::Filename_T& iODInputFilename);
00103
00107         ~AIRSCHED_Service ();
00108
00109
00110     public:
00111         // ////////////////////////////////// Business Methods //////////////////////////////////
00119         void buildSampleBom();
00120
00125         void buildSegmentPathList (stdair::TravelSolutionList_T&,
00126                                     const stdair::BookingRequestStruct&);
00127
00133         void simulate();
00134
00135
00136     public:
00137         // ////////////////////////////////// Export support methods //////////////////////////////////
00149         std::string jsonExport (const stdair::AirlineCode_T&,
00150                                 const stdair::FlightNumber_T&,
00151                                 const stdair::Date_T& iDepartureDate) const;
00152
00153
00154     public:
00155         // ////////////////////////////////// Display support methods //////////////////////////////////
00163         std::string csvDisplay () const;
00164
00178         std::string csvDisplay (const stdair::AirlineCode_T&,
00179                                 const stdair::FlightNumber_T&,
00180                                 const stdair::Date_T& iDepartureDate) const;
00181
00182
00183     private:
00184         // ////////////////////////////////// Construction and Destruction helper methods //////////////////////////////////
00188         AIRSCHED_Service ();
00189
00193         AIRSCHED_Service (const AIRSCHED_Service&);
00194

```

```

00204     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00205                                                    const stdair::BasDBParams&);
00206
00215     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&);
00216
00225     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00226                           const bool iOwnStdairService);
00227
00232     void initServiceContext();
00233
00240     void initAirschedService();
00241
00245     void finalise();
00246
00247
00248     private:
00249         // ////////// Service Context //////////
00253         AIRSCHEd_ServiceContext* _airschedServiceContext;
00254     };
00255 }
00256 #endif // __AIRSCHEd_SVC_AIRSCHEd_SERVICE_HPP

```

26.3 airsched/AIRSCHEd_Types.hpp File Reference

```

#include <boost/shared_ptr.hpp>
#include <stdair/stdair_exceptions.hpp>

```

Classes

- class [AIRSCHEd::SegmentDateNotFoundException](#)
- class [AIRSCHEd::OnDInputFileNotFoundException](#)
- class [AIRSCHEd::ScheduleInputFileNotFoundException](#)

Namespaces

- namespace [AIRSCHEd](#)

Typedefs

- typedef boost::shared_ptr< AIRSCHEd_Service > [AIRSCHEd::AIRSCHEd_ServicePtr_T](#)

26.4 AIRSCHEd_Types.hpp

```

00001 #ifndef __AIRSCHEd_AIRSCHEd_TYPES_HPP
00002 #define __AIRSCHEd_AIRSCHEd_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 AIRSCHED {
00013
00014     // Forward declarations
00015     class AIRSCHED_Service;
00016
00017
00018     // ////////// Exceptions //////////
00023     class SegmentDateNotFoundExpection : public stdair::ParserException {
00024     public:
00028         SegmentDateNotFoundExpection (const std::string& iWhat)
00029             : stdair::ParserException (iWhat) {}
00030     };
00031
00035     class OnDInputFileNotFoundExpection : public stdair::FileNotFoundExpection {
00036     public:
00040         OnDInputFileNotFoundExpection (const std::string& iWhat)
00041             : stdair::FileNotFoundExpection (iWhat) {}
00042     };
00043
00047     class ScheduleInputFileNotFoundExpection
00048         : public stdair::FileNotFoundExpection {
00049     public:
00053         ScheduleInputFileNotFoundExpection (const std::string& iWhat)
00054             : stdair::FileNotFoundExpection (iWhat) {}
00055     };
00056
00057
00058     // ////////// Type definitions specific to AirSched //////////
00062     typedef boost::shared_ptr<AIRSCHED_Service> AIRSCHED_ServicePtr_T;
00063
00064 }
00065 #endif // __AIRSCHED_AIRSCHED_TYPES_HPP

```

26.5 airsched/basic/BasConst.cpp File Reference

```

#include <airsched/basic/BasConst_General.hpp>
#include <airsched/basic/BasConst_AIRSCHED_Service.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Variables

- const int [AIRSCHED::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION](#)
= 100000

26.6 BasConst.cpp

```

00001 #include <airsched/basic/BasConst_General.hpp>

```

```

00002 #include <airsched/basic/BasConst_AIRSCHEd_Service.hpp>
00003
00004 namespace AIRSCHEd {
00005
00008     const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000;
00009
00010 }

```

26.7 airsched/basic/BasConst_AIRSCHEd_Service.hpp File Reference

Namespaces

- namespace [AIRSCHEd](#)

26.8 BasConst_AIRSCHEd_Service.hpp

```

00001 #ifndef __AIRSCHEd_BAS_BASCONST_AIRSCHEd_SERVICE_HPP
00002 #define __AIRSCHEd_BAS_BASCONST_AIRSCHEd_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace AIRSCHEd {
00009
00010 }
00011 #endif // __AIRSCHEd_BAS_BASCONST_AIRSCHEd_SERVICE_HPP

```

26.9 airsched/basic/BasConst_General.hpp File Reference

Namespaces

- namespace [AIRSCHEd](#)

26.10 BasConst_General.hpp

```

00001 #ifndef __AIRSCHEd_BAS_BASCONST_GENERAL_HPP
00002 #define __AIRSCHEd_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace AIRSCHEd {
00009
00012     extern const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION;
00013
00014 }
00015 #endif // __AIRSCHEd_BAS_BASCONST_GENERAL_HPP

```

26.11 airsched/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 [AIRSCHED](#)

Typedefs

- typedef char [AIRSCHED::char_t](#)
- typedef boost::spirit::classic::file_iterator< [char_t](#) > [AIRSCHED::iterator_t](#)
- typedef boost::spirit::classic::scanner< [iterator_t](#) > [AIRSCHED::scanner_t](#)
- typedef boost::spirit::classic::rule< [scanner_t](#) > [AIRSCHED::rule_t](#)
- typedef boost::spirit::classic::int_parser< unsigned int, 10, 1, 1 > [AIRSCHED::int1_p_t](#)
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 2, 2 > [AIRSCHED::uint2_p_t](#)
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 4, 4 > [AIRSCHED::uint4_p_t](#)
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 4 > [AIRSCHED::uint1_4_p_t](#)
- typedef boost::spirit::classic::chset< [char_t](#) > [AIRSCHED::chset_t](#)
- typedef boost::spirit::classic::impl::loop_traits< [chset_t](#), unsigned int, unsigned int >::type [AIRSCHED::repeat_p_t](#)
- typedef boost::spirit::classic::bounded< [uint2_p_t](#), unsigned int > [AIRSCHED::bounded2_p_t](#)
- typedef boost::spirit::classic::bounded< [uint4_p_t](#), unsigned int > [AIRSCHED::bounded4_p_t](#)
- typedef boost::spirit::classic::bounded< [uint1_4_p_t](#), unsigned int > [AIRSCHED::bounded1_4_p_t](#)

26.12 BasParserTypes.hpp

```

00001 #ifndef __AIRSCHED_BAS_BASCOMPARSERTYPES_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
00022
00023 // //////////////////////////////////////
00024 //
00025 // Definition of Basic Types
00026 //
00027 // //////////////////////////////////////
00028 // For a file, the parsing unit is the character (char). For a string,
00029 // it is a "char const *".
00030 // typedef char const* iterator_t;
00031 typedef char char_t;
00032
00033 // The types of iterator, scanner and rule are then derived from
00034 // the parsing unit.
00035 typedef boost::spirit::classic::file_iterator<char_t> iterator_t;
00036 typedef boost::spirit::classic::scanner<iterator_t> scanner_t;
00037 typedef boost::spirit::classic::rule<scanner_t> rule_t;
00038
00039 // //////////////////////////////////////
00040 //
00041 // Parser related types
00042 //
00043 // //////////////////////////////////////
00045 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00046
00048 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> uint2_p_t;
00049
00051 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t;
00052
00054 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
uint1_4_p_t;
00055
00057 typedef boost::spirit::classic::chset<char_t> chset_t;
00058
00061 typedef boost::spirit::classic::impl::loop_traits<chset_t,
00062 unsigned int,
00063 unsigned int>::type repeat_p_t;
00064
00066 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t;
00067 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t;

```

```

00068     typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
        bounded1_4_p_t;
00069
00070 }
00071 #endif // __AIRSCHED_BAS_BASCOMPARSERTYPES_HPP

```

26.13 airsched/batches/airsched.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 <airsched/AIRSCHED_Service.hpp>
#include <airsched/batches/BookingRequestParser.hpp>
#include <airsched/config/airsched-paths.hpp>

```

Typedefs

- typedef std::vector< std::string > [WordList_T](#)

Functions

- const std::string [K_AIRSCHED_DEFAULT_LOG_FILENAME](#) ("airsched.log")
- const std::string [K_AIRSCHED_DEFAULT_INPUT_FILENAME](#) (STDAIR_SAMPLE_ - DIR"/schedule03.csv")
- const std::string [K_AIRSCHED_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 &iolsBuiltin, 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_AIRSCHED_DEFAULT_BUILT_IN_INPUT](#) = false
- const bool [K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE](#) = false
- const int [K_AIRSCHED_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 [airsched.cpp](#).

26.13.2 Function Documentation

26.13.2.1 const std::string K_AIRSCHED_DEFAULT_LOG_FILENAME ("airsched.log")

Default name and location for the log file.

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

26.13.2.2 const std::string K_AIRSCHED_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_AIRSCHED_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 searched against the AirSched network.

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

26.13.2.4 std::string createStringFromWordList (const WordList_T & iWordList)

Definition at line 59 of file [airsched.cpp](#).

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

26.13.2.5 template<class T > std::ostream& operator<< (std::ostream & os, const std::vector< T > & v)

Definition at line 77 of file [airsched.cpp](#).

26.13.2.6 `int readConfiguration (int argc, char * argv[], bool & iolsBuiltin, 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 [airsched.cpp](#).

References [createStringFromWordList\(\)](#), [K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE](#), [K_AIRSCHED_DEFAULT_INPUT_FILENAME\(\)](#), [K_AIRSCHED_DEFAULT_LOG_FILENAME\(\)](#), and [K_AIRSCHED_EARLY_RETURN_STATUS](#).

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

26.13.2.7 `stdair::BookingRequestStruct parseBookingRequest (const std::string & iRequestOption)`

Definition at line 226 of file [airsched.cpp](#).

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

26.13.2.8 `int main (int argc, char * argv[])`

Definition at line 331 of file [airsched.cpp](#).

References [AIRSCHED::AIRSCHED_Service::buildSampleBom\(\)](#), [AIRSCHED::AIRSCHED_Service::buildSegmentPathList\(\)](#), [K_AIRSCHED_DEFAULT_BOOKING_REQUEST\(\)](#), [K_AIRSCHED_EARLY_RETURN_STATUS](#), [AIRSCHED::AIRSCHED_Service::parseAndLoad\(\)](#), [parseBookingRequest\(\)](#), and [readConfiguration\(\)](#).

26.13.3 Variable Documentation

26.13.3.1 `const bool K_AIRSCHED_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 [airsched.cpp](#).

26.13.3.2 `const bool K_AIRSCHED_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 [airsched.cpp](#).

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

26.13.3.3 `const int K_AIRSCHED_EARLY_RETURN_STATUS = 99`

Early return status (so that it can be differentiated from an error).

Definition at line 84 of file [airsched.cpp](#).

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

26.14 airsched.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 // AirSched
00019 #include <airsched/AIRSCHEDED_Service.hpp>
00020 #include <airsched/batches/BookingRequestParser.hpp>
00021 #include <airsched/config/airsched-paths.hpp>
00022
00023 // ////////// Type definitions //////////
00024 typedef std::vector<std::string> WordList_T;
00025
00026
00027 // ////////// Constants //////////
00031 const std::string K_AIRSCHEDED_DEFAULT_LOG_FILENAME ("airsched.log");
00032
00036 const std::string K_AIRSCHEDED_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00037                                                         "/schedule03.csv");
00038
00044 const bool K_AIRSCHEDED_DEFAULT_BUILT_IN_INPUT = false;
00045
00050 const bool K_AIRSCHEDED_DEFAULT_BOOKING_REQUEST_MODE = false;
00051
00056 const std::string K_AIRSCHEDED_DEFAULT_BOOKING_REQUEST ("NCE BKK NCE 2007-04-21 200
00057 7-03-21 08:32:00 C 1 DF RO 5 NONE 10:00:00 2000.0 20.0");
00058
00059 // //////////////////////////////////////
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
00083 const int K_AIRSCHED_EARLY_RETURN_STATUS = 99;
00084
00085 int readConfiguration (int argc, char* argv[],
00086                        bool& ioIsBuiltin, bool& ioReadBookingRequestFromCmdLine,
00087                        stdair::Filename_T& ioInputFilename,
00088                        std::string& ioLogFilename,
00089                        std::string& ioBookingRequestString) {
00090     // Default for the booking request mode (whether it is read from command-line)
00091     ioReadBookingRequestFromCmdLine = K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE;
00092
00093     //
00094     WordList_T lWordList;
00095
00096     // Declare a group of options that will be allowed only on command line
00097     boost::program_options::options_description generic ("Generic options");
00098     generic.add_options()
00099         ("prefix", "print installation prefix")
00100         ("version,v", "print version string")
00101         ("help,h", "produce help message");
00102
00103     // Declare a group of options that will be allowed both on command
00104     // line and in config file
00105     boost::program_options::options_description config ("Configuration");
00106     config.add_options()
00107         ("builtin,b",
00108          "The sample BOM tree can be either built-in or parsed from input files. In t
00109          hat latter case, the -i/--input option must be specified as well")
00110         ("input,i",
00111          boost::program_options::value< std::string >(&ioInputFilename)->default_valu
00112          e(K_AIRSCHED_DEFAULT_INPUT_FILENAME),
00113          "(CSV) input file specifying the schedule (flight-period) entries")
00114         ("log,l",
00115          boost::program_options::value< std::string >(&ioLogFilename)->default_value(
00116          K_AIRSCHED_DEFAULT_LOG_FILENAME),
00117          "Filename for the logs")
00118         ("read_booking_request,r",
00119          "Indicates that a booking request is given as a command-line option. That la
00120          tter must then be given with the -b/--bkg_req option")
00121         ("bkg_req,q",
00122          boost::program_options::value< WordList_T >(&lWordList)->multitoken(),
00123          "Booking request word list (e.g. 'NCE BKK NCE 2007-04-21 2007-04-21 10:00:00
00124          C 1 DF RO 5 NONE 10:0:0 2000.0 20.0')", which should be located at the end of the
00125          command line (otherwise, the other options would be interpreted as part of that
00126          booking request word list)")
00127         ;
00128
00129     // Hidden options, will be allowed both on command line and
00130     // in config file, but will not be shown to the user.
00131     boost::program_options::options_description hidden ("Hidden options");
00132     hidden.add_options()
00133         ("copyright",
00134          boost::program_options::value< std::vector<std::string> >(),
00135          "Show the copyright (license)");
00136
00137     boost::program_options::options_description cmdline_options;
00138     cmdline_options.add(generic).add(config).add(hidden);
00139
00140     boost::program_options::options_description config_file_options;

```

```

00136     config_file_options.add(config).add(hidden);
00137
00138     boost::program_options::options_description visible ("Allowed options");
00139     visible.add(generic).add(config);
00140
00141     boost::program_options::positional_options_description p;
00142     p.add ("copyright", -1);
00143
00144     boost::program_options::variables_map vm;
00145     boost::program_options::
00146         store (boost::program_options::command_line_parser (argc, argv).
00147             options (cmdline_options).positional(p).run(), vm);
00148
00149     std::ifstream ifs ("airsched.cfg");
00150     boost::program_options::store (parse_config_file (ifs, config_file_options),
00151         vm);
00152     boost::program_options::notify (vm);
00153
00154     if (vm.count ("help")) {
00155         std::cout << visible << std::endl;
00156         return K_AIRSCHED_EARLY_RETURN_STATUS;
00157     }
00158
00159     if (vm.count ("version")) {
00160         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00161         return K_AIRSCHED_EARLY_RETURN_STATUS;
00162     }
00163
00164     if (vm.count ("prefix")) {
00165         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00166         return K_AIRSCHED_EARLY_RETURN_STATUS;
00167     }
00168
00169     if (vm.count ("builtin")) {
00170         ioIsBuiltin = true;
00171     }
00172     const std::string isBuiltinStr = (ioIsBuiltin == true)? "yes": "no";
00173     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00174
00175     //
00176     std::ostringstream oErrorMessageStr;
00177     oErrorMessageStr << "Either the -b/--builtin option, or the -i/--input option"
00178         << " must be specified";
00179
00180     if (ioIsBuiltin == false) {
00181         if (vm.count ("input")) {
00182             ioInputFilename = vm["input"].as< std::string >();
00183             std::cout << "Input filename is: " << ioInputFilename << std::endl;
00184
00185         } else {
00186             // The built-in option is not selected. However, no schedule input file
00187             // is specified
00188             std::cerr << oErrorMessageStr.str() << std::endl;
00189         }
00190     }
00191
00192     //
00193     if (vm.count ("read_booking_request")) {
00194         ioReadBookingRequestFromCmdLine = true;
00195     }
00196     const std::string readBookingRequestFromCmdLineStr =
00197         (ioReadBookingRequestFromCmdLine == true)? "yes": "no";

```

```

00198     std::cout << "A booking request is to be given as command-line option? "
00199             << readBookingRequestFromCmdLineStr << std::endl;
00200
00201     if (ioReadBookingRequestFromCmdLine == true) {
00202
00203         if (lWordList.empty() == true) {
00204             std::cerr << "When the --read_booking_request/-r option is given, "
00205                     << "a query must also be provided (with the --bkg_req/-b "
00206                     << "option at the end of the command-line)" << std::endl;
00207             return K_AIRSCHED_EARLY_RETURN_STATUS;
00208         }
00209
00210         // Rebuild the booking request query string
00211         ioBookingRequestString = createStringFromWordList (lWordList);
00212         std::cout << "The booking request string is: " << ioBookingRequestString
00213             << std::endl;
00214     }
00215
00216     if (vm.count ("log")) {
00217         ioLogFilename = vm["log"].as< std::string >();
00218         std::cout << "Log filename is: " << ioLogFilename << std::endl;
00219     }
00220
00221     return 0;
00222 }
00223
00224 // //////////////////////////////////////
00225 stdair::BookingRequestStruct
00226 parseBookingRequest (const std::string& iRequestOption) {
00227     typedef boost::tokenizer<boost::char_separator<char> > tokenizer;
00228     boost::char_separator<char> sep(" -:");
00229
00230     tokenizer tokens (iRequestOption, sep);
00231
00232     // Origin (e.g., "NCE")
00233     tokenizer::iterator tok_iter = tokens.begin();
00234     assert (tok_iter != tokens.end());
00235     const stdair::AirportCode_T iOrigin (*tok_iter);
00236
00237     // Destination (e.g., "BKK")
00238     ++tok_iter; assert (tok_iter != tokens.end());
00239     const stdair::AirportCode_T iDestination (*tok_iter);
00240
00241     // POS (e.g., "NCE")
00242     ++tok_iter; assert (tok_iter != tokens.end());
00243     const stdair::AirportCode_T iPOS (*tok_iter);
00244
00245     // Preferred departure date (e.g., "2007-04-21")
00246     ++tok_iter; assert (tok_iter != tokens.end());
00247     const short lDepDateYear = boost::lexical_cast<short> (*tok_iter);
00248     ++tok_iter; assert (tok_iter != tokens.end());
00249     const short lDepDateMonth = boost::lexical_cast<short> (*tok_iter);
00250     ++tok_iter; assert (tok_iter != tokens.end());
00251     const short lDepDateDay = boost::lexical_cast<short> (*tok_iter);
00252     const stdair::Date_T iDepartureDate(lDepDateYear, lDepDateMonth, lDepDateDay);
00253
00254     // Request date (e.g., "2007-03-21")
00255     ++tok_iter; assert (tok_iter != tokens.end());
00256     const short lReqDateYear = boost::lexical_cast<short> (*tok_iter);
00257     ++tok_iter; assert (tok_iter != tokens.end());
00258     const short lReqDateMonth = boost::lexical_cast<short> (*tok_iter);
00259     ++tok_iter; assert (tok_iter != tokens.end());

```



```

00260     const short lReqDateDay = boost::lexical_cast<short> (*tok_iter);
00261     const stdair::Date_T iRequestDate (lReqDateYear, lReqDateMonth, lReqDateDay);
00262
00263     // Request time (e.g., "08:34:23")
00264     ++tok_iter; assert (tok_iter != tokens.end());
00265     const short lReqTimeHours = boost::lexical_cast<short> (*tok_iter);
00266     ++tok_iter; assert (tok_iter != tokens.end());
00267     const short lReqTimeMinutes = boost::lexical_cast<short> (*tok_iter);
00268     ++tok_iter; assert (tok_iter != tokens.end());
00269     const short lReqTimeSeconds = boost::lexical_cast<short> (*tok_iter);
00270     const stdair::Duration_T iRequestTime (lReqTimeHours, lReqTimeMinutes,
00271                                           lReqTimeSeconds);
00272
00273     // Request date-time (aggregation of the two items above)
00274     const stdair::DateTime_T iRequestDateTime (iRequestDate, iRequestTime);
00275
00276     // Preferred cabin (e.g., "C")
00277     ++tok_iter; assert (tok_iter != tokens.end());
00278     const stdair::CabinCode_T iPreferredCabin (*tok_iter);
00279
00280     // Party size (e.g., 1)
00281     ++tok_iter; assert (tok_iter != tokens.end());
00282     const stdair::NbOfSeats_T iPartySize = 1;
00283
00284     // Channel (e.g., "DF")
00285     ++tok_iter; assert (tok_iter != tokens.end());
00286     const stdair::ChannelLabel_T iChannel (*tok_iter);
00287
00288     // Trip type (e.g., "RO")
00289     ++tok_iter; assert (tok_iter != tokens.end());
00290     const stdair::TripType_T iTripType (*tok_iter);
00291
00292     // Stay duration (e.g., 5)
00293     ++tok_iter; assert (tok_iter != tokens.end());
00294     const stdair::DayDuration_T iStayDuration = 5;
00295
00296     // Frequent flyer (e.g., "NONE")
00297     ++tok_iter; assert (tok_iter != tokens.end());
00298     const stdair::FrequentFlyer_T iFrequentFlyerType ("NONE");
00299
00300     // Preferred departure time (e.g., "10:00:00")
00301     ++tok_iter; assert (tok_iter != tokens.end());
00302     const short lPrefTimeHours = boost::lexical_cast<short> (*tok_iter);
00303     ++tok_iter; assert (tok_iter != tokens.end());
00304     const short lPrefTimeMinutes = boost::lexical_cast<short> (*tok_iter);
00305     ++tok_iter; assert (tok_iter != tokens.end());
00306     const short lPrefTimeSeconds = boost::lexical_cast<short> (*tok_iter);
00307     const stdair::Duration_T iPreferredDepartureTime (lPrefTimeHours,
00308                                                       lPrefTimeMinutes,
00309                                                       lPrefTimeSeconds);
00310
00311     // Willingness-to-pay (e.g., 2000.0)
00312     ++tok_iter; assert (tok_iter != tokens.end());
00313     const stdair::WTP_T iWTP = 2000.0;
00314
00315     // Value of time (e.g., 20.0)
00316     ++tok_iter; assert (tok_iter != tokens.end());
00317     const stdair::PriceValue_T iValueOfTime = 20.0;
00318
00319     // Build and return the booking request structure
00320     return stdair::BookingRequestStruct (iOrigin,
00321                                         iDestination, iPOS,

```

```

00322         iDepartureDate, iRequestDateTime,
00323         iPreferredCabin, iPartySize,
00324         iChannel, iTripType, iStayDuration,
00325         iFrequentFlyerType,
00326         iPreferredDepartureTime, iWTP,
00327         iValueOfTime);
00328     }
00329
00330     // //////////// M A I N ////////////
00331     int main (int argc, char* argv[]) {
00332
00333         // State whether the BOM tree should be built-in or parsed from an
00334         // input file
00335         bool isBuiltin;
00336
00337         // A booking request should be given as command-line option
00338         bool readBookingRequestFromCmdLine;
00339
00340         // Input file name
00341         stdair::Filename_T lInputFilename;
00342
00343         // Output log File
00344         stdair::Filename_T lLogFilename;
00345
00346         // Booking request string
00347         std::string lBookingRequestString;
00348
00349         // Call the command-line option parser
00350         const int lOptionParserStatus =
00351             readConfiguration (argc, argv, isBuiltin, readBookingRequestFromCmdLine,
00352                               lInputFilename, lLogFilename, lBookingRequestString);
00353
00354         if (lOptionParserStatus == K_AIRSCHED_EARLY_RETURN_STATUS) {
00355             return 0;
00356         }
00357
00358         // Set the log parameters
00359         std::ofstream logOutputFile;
00360         // Open and clean the log outputfile
00361         logOutputFile.open (lLogFilename.c_str());
00362         logOutputFile.clear();
00363
00364         // Initialise the AirSched service object
00365         const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00366         AIRSCHED::AIRSCHED_Service airschedService (lLogParams);
00367
00368         // Check wether or not (CSV) input files should be read
00369         if (isBuiltin == true) {
00370
00371             // Build the sample BOM tree
00372             airschedService.buildSampleBom();
00373
00374         } else {
00375             // Build the BOM tree from parsing input files
00376             airschedService.parseAndLoad (lInputFilename);
00377         }
00378
00379         // Check wether or not a booking request is given as a command-line option
00380         if (readBookingRequestFromCmdLine == false) {
00381             lBookingRequestString = K_AIRSCHED_DEFAULT_BOOKING_REQUEST;
00382         }
00383     }

```

```
00384 // DEBUG
00385 STDAIR_LOG_DEBUG("Booking request string: '" << lBookingRequestString << "'");
00386
00387 // Create a booking request object
00388 const stdair::BookingRequestStruct& lBookingRequest =
00389     parseBookingRequest (lBookingRequestString);
00390
00391 //
00392 stdair::TravelSolutionList_T lTravelSolutionList;
00393 airschedService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
00394
00395 // DEBUG
00396 STDAIR_LOG_DEBUG ("Parsed booking request: " << lBookingRequest);
00397
00398 // DEBUG
00399 std::ostream oStream;
00400 stdair::BomDisplay::csvDisplay (oStream, lTravelSolutionList);
00401 STDAIR_LOG_DEBUG (oStream.str());
00402
00403 // Close the Log outputFile
00404 logOutputFile.close();
00405
00406 return 0;
00407 }
```

26.15 airsched/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 <airsched/batches/BookingRequestParser.hpp>
```

Classes

- struct [airsched::store_place_element](#)
- struct [airsched::store_date](#)
- struct [airsched::store_airline_sign](#)
- struct [airsched::store_airline_code](#)
- struct [airsched::store_airline_name](#)
- struct [airsched::store_passenger_number](#)
- struct [airsched::store_adult_passenger_type](#)
- struct [airsched::store_child_passenger_type](#)
- struct [airsched::store_pet_passenger_type](#)
- struct [airsched::SearchStringParser](#)
- struct [airsched::SearchStringParser::definition< ScannerT >](#)

Namespaces

- namespace [airsched](#)

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 [airsched::parseBookingRequest](#) (const std::string &iSearchString)

Variables

- boost::spirit::classic::int_parser< unsigned int, 10, 1, 1 > [airsched::int1_p](#)
- boost::spirit::classic::uint_parser< unsigned int, 10, 1, 1 > [airsched::uint1_p](#)
- boost::spirit::classic::uint_parser< unsigned int, 10, 1, 2 > [airsched::uint1_2_p](#)
- boost::spirit::classic::uint_parser< int, 10, 2, 2 > [airsched::uint2_p](#)
- boost::spirit::classic::uint_parser< int, 10, 2, 4 > [airsched::uint2_4_p](#)
- boost::spirit::classic::uint_parser< int, 10, 4, 4 > [airsched::uint4_p](#)
- boost::spirit::classic::uint_parser< int, 10, 1, 4 > [airsched::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 // AirSched
00025 #include <airsched/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 airsched {
00035
00036     struct store_place_element {
00037     store_place_element (SearchString_T& ioSearchString)
00040         : _searchString (ioSearchString) {}
00041
00042     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00043         std::string lPlace (iStr, iStrEnd);
00044         // std::cout << "Place: " << lPlace << std::endl;
00045
00046         // Set the place
00047         _searchString._tmpPlace._name += " " + lPlace;
00048
00049         // Add the parsed place to the list
00050         // _searchString._placeList.push_back (_searchString._tmpPlace);
00051     }
00052
00053     SearchString_T& _searchString;
00054 };
00055
00056     struct store_date {
00057     store_date (SearchString_T& ioSearchString)
00058         : _searchString (ioSearchString) {}
00059
00060     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00061         _searchString._tmpDate._date = _searchString._tmpDate.getDate();
00062         // std::cout << "Board date: "
00063         // << _searchString._date << std::endl;
00064
00065         // Add the parsed date to the list
00066         _searchString._dateList.push_back (_searchString._tmpDate);
00067     }
00068
00069     SearchString_T& _searchString;
00070 };
00071
00072     struct store_airline_sign {
00073     store_airline_sign (SearchString_T& ioSearchString)
00074         : _searchString (ioSearchString) {}
00075
00076     void operator() (bool iAirlineSign) const {
00077         _searchString._tmpAirline._isPreferred = !iAirlineSign;
00078         // std::cout << "Airline is preferred: " << iAirlineSign << std::endl;
00079     }
00080
00081     SearchString_T& _searchString;
00082 };
00083
00084     struct store_airline_code {
00085     store_airline_code (SearchString_T& ioSearchString)
00086         : _searchString (ioSearchString) {}
00087
00088     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00089         std::string lAirlineCode (iStr, iStrEnd);
00090         _searchString._tmpAirline._code = lAirlineCode;
00091         // std::cout << "Airline code: " << lAirlineCode << std::endl;
00092
00093         // Add the parsed airline to the list

```

```

00104     _searchString._airlineList.push_back (_searchString._tmpAirline);
00105 }
00106
00107 SearchString_T& _searchString;
00108 };
00109
00111 struct store_airline_name {
00113     store_airline_name (SearchString_T& ioSearchString)
00114         : _searchString (ioSearchString) {}
00115
00117     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00118         std::string lAirlineName (iStr, iStrEnd);
00119         _searchString._tmpAirline._name = lAirlineName;
00120         // std::cout << "Airline: " << lAirlineName << std::endl;
00121
00122         // Add the parsed airline to the list
00123         _searchString._airlineList.push_back (_searchString._tmpAirline);
00124     }
00125
00126     SearchString_T& _searchString;
00127 };
00128
00130 struct store_passenger_number {
00132     store_passenger_number (SearchString_T& ioSearchString)
00133         : _searchString (ioSearchString) {}
00134
00136     void operator() (unsigned int iNumber) const {
00137         _searchString._tmpPassenger._number = iNumber;
00138         // std::cout << "Number of passengers: " << iNumber << std::endl;
00139     }
00140
00141     SearchString_T& _searchString;
00142 };
00143
00145 struct store_adult_passenger_type {
00147     store_adult_passenger_type (SearchString_T& ioSearchString)
00148         : _searchString (ioSearchString) {}
00149
00151     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00152         std::string lPassengerType (iStr, iStrEnd);
00153         _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
00164 struct store_child_passenger_type {
00166     store_child_passenger_type (SearchString_T& ioSearchString)
00167         : _searchString (ioSearchString) {}
00168
00170     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00171         std::string lPassengerType (iStr, iStrEnd);
00172         _searchString._tmpPassenger._type = Passenger_T::CHILD;
00173         // std::cout << "Passenger type: " << lPassengerType << std::endl;
00174
00175         // Add the parsed passenger to the list
00176         _searchString._passengerList.push_back (_searchString._tmpPassenger);
00177     }

```

```

00178
00179     SearchString_T& _searchString;
00180 };
00181
00182 struct store_pet_passenger_type {
00183     store_pet_passenger_type (SearchString_T& ioSearchString)
00184         : _searchString (ioSearchString) {}
00185
00186     void operator() (iterator_t iStr, iterator_t iStrEnd) const {
00187         std::string lPassengerType (iStr, iStrEnd);
00188         _searchString._tmpPassenger._type = Passenger_T::PET;
00189         // std::cout << "Passenger type: " << lPassengerType << std::endl;
00190
00191         // Add the parsed passenger to the list
00192         _searchString._passengerList.push_back (_searchString._tmpPassenger);
00193     }
00194
00195     SearchString_T& _searchString;
00196 };
00197
00198 // ////////////////////////////////// Utilities //////////////////////////////////
00199 boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p;
00200 boost::spirit::classic::uint_parser<unsigned int, 10, 1, 1> uint1_p;
00201 boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> uint1_2_p;
00202 boost::spirit::classic::uint_parser<int, 10, 2, 2> uint2_p;
00203 boost::spirit::classic::uint_parser<int, 10, 2, 4> uint2_4_p;
00204 boost::spirit::classic::uint_parser<int, 10, 4, 4> uint4_p;
00205 boost::spirit::classic::uint_parser<int, 10, 1, 4> uint1_4_p;
00206
00207 //
00208 // Our calculator grammar (using subrules)
00209 //
00210 using namespace boost::spirit::classic;
00211
00212 struct SearchStringParser :
00213     public boost::spirit::classic::grammar<SearchStringParser> {
00214     SearchStringParser (SearchString_T& ioSearchString)
00215         : _searchString (ioSearchString) {}
00216
00217     template <typename ScannerT>
00218     struct definition {
00219         definition (SearchStringParser const& self) {
00220
00221             search_string = places
00222                 >> !( dates )
00223                 >> *( preferred_airlines )
00224                 >> *( passengers )
00225             ;
00226
00227             places =
00228                 +( place_element )
00229             ;
00230
00231             place_element =
00232                 lexeme_d[ (repeat_p(1,20)[chset_p("a-z")]) [store_place_element(self._searchString)] ]
00233             ;
00234
00235             dates =

```



```

00277         date[store_date(self._searchString)]
00278         >> !date[store_date(self._searchString)]
00279         ;
00280
00281         date =
00282         ( month | day )
00283         >> boost::spirit::classic::chset_p("/-")
00284         >> ( day | month )
00285         >> ! ( boost::spirit::classic::chset_p("/-")
00286                 >> year )
00287         ;
00288
00289         day =
00290         lexeme_d[ limit_d(1u,31u) [uint1_2_p] [assign_a(self._searchString._tmpDa
00291 te._day)] ]
00292         ;
00293         month =
00294         lexeme_d[ limit_d(1u,12u) [uint1_2_p] [assign_a(self._searchString._tmpDa
00295 te._month)] ]
00296         ;
00297         year =
00298         lexeme_d[ limit_d(2000u,2099u) [uint4_p] [assign_a(self._searchString._tm
00299 pDate._year)] ]
00299         | lexeme_d[ limit_d(0u,99u) [uint2_p] [assign_a(self._searchString._tmpDa
00300 te._year)] ]
00301         ;
00302         preferred_airlines =
00303         !(boost::spirit::classic::sign_p) [store_airline_sign(self._searchString
00304 )]
00305         >> airline_code | airline_name
00306         ;
00307         airline_code =
00308         lexeme_d[ (repeat_p(2,3) [chset_p("0-9a-z")]) [store_airline_code(self._s
00309 earchString)] ]
00310         ;
00311         airline_name =
00312         lexeme_d[ (repeat_p(4,20) [chset_p("0-9a-z")]) [store_airline_name(self._
00313 searchString)] ]
00314         ;
00315         passengers =
00316         passenger_number >> passenger_type
00317         ;
00318
00319         passenger_number =
00320         lexeme_d[ limit_d(1u, 9u) [uint1_p] [store_passenger_number(self._searchS
00321 tring)] ]
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 airsched/batches/BookingRequestParser.hpp File Reference

```
#include <string>
```

```
#include <vector>
```

Classes

- struct [airsched::Place_T](#)
- struct [airsched::Date_T](#)
- struct [airsched::Airline_T](#)
- struct [airsched::Passenger_T](#)
- struct [airsched::SearchString_T](#)

Namespaces

- namespace [airsched](#)

Typedefs

- typedef std::vector< Place_T > [airsched::PlaceList_T](#)
- typedef std::vector< Date_T > [airsched::DateList_T](#)
- typedef std::vector< Airline_T > [airsched::AirlineList_T](#)
- typedef std::vector< Passenger_T > [airsched::PassengerList_T](#)

Functions

- SearchString_T [airsched::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 airsched {
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                 << std::endl;
00064         }
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;
00078         Passenger_T () : _type(ADULT), _number(1) {}
00079         /* Display. */
00080         void display() const {

```

```

00081         std::cout << "Passenger: " << _number << " (" << _labels[_type] << ")"
00082         << std::endl;
00083     }
00084 };
00085
00087     const std::string Passenger_T::_labels[Passenger_T::LAST_VALUE] =
00088     { "Adult", "Child", "Pet" };
00089
00091     typedef std::vector<Passenger_T> PassengerList_T;
00092
00094     struct SearchString_T {
00095         // Attributes
00096         PlaceList_T _placeList;
00097         DateList_T _dateList;
00098         AirlineList_T _airlineList;
00099         PassengerList_T _passengerList;
00100
00102         SearchString_T () {}
00103
00104         /* Display. */
00105         void display() const {
00106             std::cout << std::endl;
00107
00108             for (PlaceList_T::const_iterator itPlace = _placeList.begin();
00109                  itPlace != _placeList.end(); ++itPlace) {
00110                 const Place_T& lPlace = *itPlace;
00111                 lPlace.display();
00112             }
00113
00114             for (DateList_T::const_iterator itDate = _dateList.begin();
00115                  itDate != _dateList.end(); ++itDate) {
00116                 const Date_T& lDate = *itDate;
00117                 lDate.display();
00118             }
00119
00120             for (AirlineList_T::const_iterator itAirline = _airlineList.begin();
00121                  itAirline != _airlineList.end(); ++itAirline) {
00122                 const Airline_T& lAirline = *itAirline;
00123                 lAirline.display();
00124             }
00125
00126             for (PassengerList_T::const_iterator itPassenger = _passengerList.begin();
00127                  itPassenger != _passengerList.end(); ++itPassenger) {
00128                 const Passenger_T& lPassenger = *itPassenger;
00129                 lPassenger.display();
00130             }
00131
00132             std::cout << "-- Staging --" << std::endl;
00133             _tmpPlace.display();
00134         }
00135
00136         // //// Staging ////
00137         Place_T _tmpPlace;
00138         Date_T _tmpDate;
00139         Airline_T _tmpAirline;
00140         Passenger_T _tmpPassenger;
00141     };
00142
00144     //
00145     // The booking request grammar (using subrules)
00146     //
00148

```

```

00174     SearchString_T parseBookingRequest (const std::string& iSearchString);
00175
00176 }

```

26.19 airsched/bom/AirportList.hpp File Reference

```

#include <set>
#include <vector>
#include <stdair/stdair_basic_types.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Typedefs

- typedef std::set< stdair::AirportCode_T > [AIRSCHED::AirportList_T](#)
- typedef std::vector< stdair::AirportCode_T > [AIRSCHED::AirportOrderedList_T](#)

26.20 AirportList.hpp

```

00001 #ifndef __AIRSCHED_BOM_AIRPORTLIST_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
00014
00016     typedef std::set<stdair::AirportCode_T> AirportList_T;
00017     typedef std::vector<stdair::AirportCode_T> AirportOrderedList_T;
00018
00019 }
00020 #endif // __AIRSCHED_BOM_AIRPORTLIST_HPP

```

26.21 airsched/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 <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/BomDisplay.hpp>
```

Classes

- struct [AIRSCHED::FlagSaver](#)

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00012 #include <airsched/bom/ReachableUniverse.hpp>
00013 #include <airsched/bom/BomDisplay.hpp>
00014
00015 namespace AIRSCHED {
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 }
00092
00093
00094 }

```

26.23 airsched/bom/BomDisplay.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
```

Classes

- class [AIRSCHED::BomDisplay](#)
Utility class to display AirSched objects with a pretty format.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.24 BomDisplay.hpp

```

00001 #ifndef __AIRSCHED_BOM_BOMDISPLAY_HPP
00002 #define __AIRSCHED_BOM_BOMDISPLAY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // AirSched
00011
00013 namespace stdair {
00014     class BomRoot;
00015 }
00016
00017 namespace AIRSCHED {
00018
00020     class ReachableUniverse;
00021
00026     class BomDisplay {
00027     public:
00028         // ////////////////////////////////// Display support methods //////////////////////////////////
00037         static std::string csvDisplay (const stdair::BomRoot&);
00038
00047         static void csvDisplay (std::ostream&, const ReachableUniverse&);
00048     };
00049
00050 }
00051 #endif // __AIRSCHED_BOM_BOMDISPLAY_HPP

```

26.25 airsched/bom/FareFamilyStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <airsched/bom/FareFamilyStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

26.26 FareFamilyStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // AIRSCHED
00008 #include <airsched/bom/FareFamilyStruct.hpp>
00009
00010 namespace AIRSCHED {

```

```

00011
00012 // //////////////////////////////////////
00013 FareFamilyStruct::
00014 FareFamilyStruct (const stdair::FamilyCode_T& iFamilyCode,
00015                  const stdair::ClassList_String_T& iClasses)
00016     : _familyCode (iFamilyCode),
00017       _classes (iClasses) {
00018 }
00019
00020 // //////////////////////////////////////
00021 const std::string FareFamilyStruct::describe() const {
00022     std::ostringstream ostr;
00023     ostr << "          " << _familyCode << " " << _classes << ", ";
00024     return ostr.str();
00025 }
00026
00027 }

```

26.27 airsched/bom/FareFamilyStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [AIRSCHED::FareFamilyStruct](#)

Namespaces

- namespace [AIRSCHED](#)

Typedefs

- typedef std::vector< FareFamilyStruct > [AIRSCHED::FareFamilyStructList_T](#)

26.28 FareFamilyStruct.hpp

```

00001 #ifndef __AIRSCHED_BOM_FAREFAMILYSTRUCT_HPP
00002 #define __AIRSCHED_BOM_FAREFAMILYSTRUCT_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
00014 namespace AIRSCHED {
00015
00017     struct FareFamilyStruct : public stdair::StructAbstract {
00018         // Attributes
00019         stdair::FamilyCode_T _familyCode;
00020         stdair::ClassList_String_T _classes;
00021
00023         FareFamilyStruct (const stdair::FamilyCode_T&,
00024                         const stdair::ClassList_String_T&);
00025
00027         const std::string describe() const;
00028     };
00029
00031     typedef std::vector<FareFamilyStruct> FareFamilyStructList_T;
00032
00033 }
00034 #endif // __AIRSCHED_BOM_FAREFAMILYSTRUCT_HPP

```

26.29 airsched/bom/FlightPeriodStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <airsched/AIRSCHED_Types.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00011 #include <airsched/AIRSCHED_Types.hpp>
00012 #include <airsched/bom/FlightPeriodStruct.hpp>
00013
00014 namespace AIRSCHED {
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         throw SegmentDateNotFoundException (oStr.str());
00134     }
00135
00136     // Add the Cabin structure to the Segment Cabin structure.
00137     assert (itSegment != _segmentList.end());
00138     SegmentStruct& lSegment = *itSegment;
00139     lSegment._cabinList.push_back (iCabin);
00140 }
00141
00142
00143
00144
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;
00205         ostr << "Within the schedule input file, there is a flight "
00206             << "for which the cabin code does not exist.";
00207         STDAIR_LOG_ERROR (ostr.str());
00208         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 airsched/bom/FlightPeriodStruct.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airsched/bom/LegCabinStruct.hpp>
#include <airsched/bom/LegStruct.hpp>

```

```
#include <airsched/bom/SegmentStruct.hpp>
#include <airsched/bom/SegmentCabinStruct.hpp>
#include <airsched/bom/FareFamilyStruct.hpp>
#include <airsched/bom/AirportList.hpp>
```

Classes

- struct [AIRSCHED::FlightPeriodStruct](#)

Namespaces

- namespace [AIRSCHED](#)

26.32 FlightPeriodStruct.hpp

```
00001 #ifndef __AIRSCHED_BOM_FLIGHTPERIODSTRUCT_HPP
00002 #define __AIRSCHED_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 // AirSched
00014 #include <airsched/bom/LegCabinStruct.hpp>
00015 #include <airsched/bom/LegStruct.hpp>
00016 #include <airsched/bom/SegmentStruct.hpp>
00017 #include <airsched/bom/SegmentCabinStruct.hpp>
00018 #include <airsched/bom/FareFamilyStruct.hpp>
00019 #include <airsched/bom/AirportList.hpp>
00020
00021 namespace AIRSCHED {
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 // __AIRSCHED_BOM_FLIGHTPERIODSTRUCT_HPP

```

26.33 airsched/bom/LegCabinStruct.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/bom/LegCabin.hpp>

#include <airsched/bom/LegCabinStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AIRSCHED
00010 #include <airsched/bom/LegCabinStruct.hpp>
00011
00012 namespace AIRSCHED {
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 airsched/bom/LegCabinStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [AIRSCHED::LegCabinStruct](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

Typedefs

- typedef std::vector< LegCabinStruct > [AIRSCHED::LegCabinStructList_T](#)

26.36 LegCabinStruct.hpp

```

00001 #ifndef __AIRSCHED_BOM_LEGCABINSTRUCT_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
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 // __AIRSCHED_BOM_LEGCABINSTRUCT_HPP

```

26.37 airsched/bom/LegStruct.cpp File Reference

```

#include <cassert>
#include <sstream>

#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airsched/bom/LegStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AIRSCHED
00011 #include <airsched/bom/LegStruct.hpp>
00012
00013 namespace AIRSCHED {
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

```

```

00065     // Set the Elapsed Time
00066     ioLegDate.setElapsedTime (_elapsed);
00067 }
00068
00069 }
```

26.39 airsched/bom/LegStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airsched/bom/LegCabinStruct.hpp>
```

Classes

- struct [AIRSCHED::LegStruct](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

Typedefs

- typedef std::vector< LegStruct > [AIRSCHED::LegStructList_T](#)

26.40 LegStruct.hpp

```

00001 #ifndef __AIRSCHED_BOM_LEGSTRUCT_HPP
00002 #define __AIRSCHED_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 // AirSched
00014 #include <airsched/bom/LegCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class LegDate;
```

```

00019 }
00020
00021 namespace AIRSCHED {
00022
00024     struct LegStruct : public stdair::StructAbstract {
00025         // Attributes
00026         stdair::AirportCode_T _boardingPoint;
00027         stdair::DateOffset_T _boardingDateOffset;
00028         stdair::Duration_T _boardingTime;
00029         stdair::AirportCode_T _offPoint;
00030         stdair::DateOffset_T _offDateOffset;
00031         stdair::Duration_T _offTime;
00032         stdair::Duration_T _elapsed;
00033         LegCabinStructList_T _cabinList;
00034
00040         void fill (const stdair::Date_T& iRefDate, stdair::LegDate&) const;
00041
00043         const std::string describe() const;
00044
00046         LegStruct();
00047     };
00048
00050     typedef std::vector<LegStruct> LegStructList_T;
00051
00052 }
00053 #endif // __AIRSCHED_BOM_LEGSTRUCT_HPP

```

26.41 airsched/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 <airsched/bom/OnDPeriodStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AIRSCHED
00013 #include <airsched/bom/OnDPeriodStruct.hpp>
00014
00015 namespace AIRSCHED {
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_NBOFAIRLINES),
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

```

```
00071 }
```

26.43 airsched/bom/OnDPeriodStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [AIRSCHED::OnDPeriodStruct](#)

Namespaces

- namespace [AIRSCHED](#)

26.44 OnDPeriodStruct.hpp

```
00001 #ifndef __AIRSCHED_BOM_ONDPERIODSTRUCT_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
00015     struct OnDPeriodStruct : public stdair::StructAbstract {
00016     public:
00017         // //////////// Getters ////////////
00019         const stdair::AirlineCode_T& getFirstAirlineCode () const;
00020
00022         stdair::Date_T getDate() const;
00023
00025         stdair::Duration_T getTime() const;
00026
00027         // //////////// Display Methods ////////////
00029         const std::string describe() const;
00030
00033         const std::string describeTSKey() const;
00034
00035     public:
00037         OnDPeriodStruct ();
00038
00039     public:
00040         // Attributes
00041         stdair::AirportCode_T _origin;
00042         stdair::AirportCode_T _destination;
00043         stdair::DatePeriod_T _datePeriod;
```



```

00044     stdair::Duration_T _timeRangeStart;
00045     stdair::Duration_T _timeRangeEnd;
00046     stdair::NbOfAirlines_T _nbOfAirlines;
00047     stdair::AirlineCode_T _airlineCode;
00048     stdair::ClassCode_T _classCode;
00049     stdair::AirlineCodeList_T _airlineCodeList;
00050     stdair::ClassCodeList_T _classCodeList;
00051
00053     stdair::Date_T _dateRangeStart;
00054     stdair::Date_T _dateRangeEnd;
00055     unsigned int _itYear;
00056     unsigned int _itMonth;
00057     unsigned int _itDay;
00058
00060     long _itHours;
00061     long _itMinutes;
00062     long _itSeconds;
00063 };
00064 }
00065 #endif // __AIRSCHED_BOM_ONDPERIODSTRUCT_HPP

```

26.45 airsched/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 <airsched/bom/OriginDestinationSet.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Functions

- template void [AIRSCHED::OriginDestinationSet::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [AIRSCHED::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 // AirSched
00014 #include <airsched/bom/OriginDestinationSet.hpp>
00015
00016 namespace AIRSCHED {
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::serialisationImplementation() {
00048     std::ostringstream oStr;
00049     boost::archive::text_oarchive oa (oStr);
00050     oa << *this;
00051
00052     std::istringstream iStr;
00053     boost::archive::text_iarchive ia (iStr);
00054     ia >> *this;
00055 }
00056
00057 // //////////////////////////////////////
00058 template<class Archive>
00059 void OriginDestinationSet::serialize (Archive& ioArchive,
00060                                     const unsigned int iFileVersion) {
00061     ioArchive & _key;
00062 }
00063
00064 // //////////////////////////////////////
00065 // Explicit template instantiation
00066 namespace ba = boost::archive;

```

```

00067     template
00068     void OriginDestinationSet::serialize<ba::text_oarchive> (ba::text_oarchive&,
00069                                                             unsigned int);
00070     template
00071     void OriginDestinationSet::serialize<ba::text_iarchive> (ba::text_iarchive&,
00072                                                             unsigned int);
00073     // //////////////////////////////////////
00074 }
00075
00076

```

26.47 airsched/bom/OriginDestinationSet.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <airsched/bom/OriginDestinationSetKey.hpp>
#include <airsched/bom/OriginDestinationSetTypes.hpp>

```

Classes

- class [AIRSCHED::OriginDestinationSet](#)
Class representing a simple sub-network.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.48 OriginDestinationSet.hpp

```

00001 #ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSET_HPP
00002 #define __AIRSCHED_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 // AirSched
00013 #include <airsched/bom/OriginDestinationSetKey.hpp>

```

```

00014 #include <airsched/bom/OriginDestinationSetTypes.hpp>
00015
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00024 namespace stdair {
00025     template <typename BOM> class FacBom;
00026     class FacBomManager;
00027 }
00028
00029 namespace AIRSCHEDED {
00030
00044     class OriginDestinationSet : public stdair::BomAbstract {
00048         template <typename BOM> friend class stdair::FacBom;
00049         friend class stdair::FacBomManager;
00050         friend class boost::serialization::access;
00051
00052     public:
00053         // ////////// Type definitions //////////
00057         typedef OriginDestinationSetKey Key_T;
00058
00059
00060     public:
00061         // ////////// Getters //////////
00065         const Key_T& getKey() const {
00066             return _key;
00067         }
00068
00072         const stdair::AirportCode_T& getDestination() const {
00073             return _key.getOffPoint();
00074         }
00075
00079         stdair::BomAbstract* const getParent() const {
00080             return _parent;
00081         }
00082
00086         const stdair::HolderMap_T& getHolderMap() const {
00087             return _holderMap;
00088         }
00089
00090
00091     public:
00092         // ////////// Display support methods //////////
00098         void toStream (std::ostream& ioOut) const {
00099             ioOut << toString();
00100         }
00101
00107         void fromStream (std::istream& ioIn) {
00108         }
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:
00132     void serialisationImplementation();
00133
00134
00135     protected:
00136     // ////////// Constructors and destructors //////////
00137     OriginDestinationSet (const Key_T&);
00138
00139     ~OriginDestinationSet();
00140
00141     private:
00142     OriginDestinationSet();
00143
00144     OriginDestinationSet (const OriginDestinationSet&);
00145
00146     protected:
00147     // ////////// Attributes //////////
00148     Key_T _key;
00149
00150     stdair::BomAbstract* _parent;
00151
00152     stdair::HolderMap_T _holderMap;
00153 };
00154 }
00155 #endif // __AIRSCHED_BOM_ORIGINDESTINATIONSET_HPP
00156

```

26.49 airsched/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 <airsched/bom/OriginDestinationSetKey.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Functions

- template void [AIRSCHED::OriginDestinationSetKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [AIRSCHED::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 // AirSched
00014 #include <airsched/bom/OriginDestinationSetKey.hpp>
00015
00016 namespace AIRSCHED {
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::serialisationImplementation() {
00058     std::ostringstream oStr;
00059     boost::archive::text_oarchive oa (oStr);
00060     oa << *this;

```

```

00061
00062     std::istringstream iStr;
00063     boost::archive::text_iarchive ia (iStr);
00064     ia >> *this;
00065 }
00066
00067 // //////////////////////////////////////
00068 template<class Archive>
00069 void OriginDestinationSetKey::serialize (Archive& ioArchive,
00070                                         const unsigned int iFileVersion) {
00071     ioArchive & _destination;
00072 }
00073 // //////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template
00077 void OriginDestinationSetKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00078
00079                                                         unsigned int);
00080 template
00081 void OriginDestinationSetKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00082
00083                                                         unsigned int);
00084 // //////////////////////////////////////
00085 }

```

26.51 airsched/bom/OriginDestinationSetKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [AIRSCHED::OriginDestinationSetKey](#)
Structure representing the key of a sub-network.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [AIRSCHED](#)

26.52 OriginDestinationSetKey.hpp

```

00001 #ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSETKEY_HPP

```

```

00002 #define __AIRSCHEDED_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 AIRSCHEDED {
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
00057     public:
00058         // ////////////////////////////////// Getters //////////////////////////////////
00062         const stdair::AirportCode_T& getOffPoint() const {
00063             return _destination;
00064         }
00065
00066
00067     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
00095     public:
00096         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00100         template<class Archive>
00101         void serialize (Archive& ar, const unsigned int iFileVersion);
00102
00103     private:
00108         void serialisationImplementation();
00109
00110
00111     private:
00112         // ////////////////////////////////// Attributes //////////////////////////////////

```



```

00116     stdair::AirportCode_T _destination;
00117 };
00118
00119 }
00120 #endif // __AIRSCHED_BOM_ORIGINDESTINATIONSETKEY_HPP

```

26.53 airsched/bom/OriginDestinationSetTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Typedefs

- typedef std::list< OriginDestinationSet * > [AIRSCHED::OriginDestinationSetList_T](#)
- typedef std::map< const stdair::MapKey_T, OriginDestinationSet * > [AIRSCHED::OriginDestinationSetMap_T](#)

26.54 OriginDestinationSetTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSETTYPES_HPP
00003 #define __AIRSCHED_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 AIRSCHED {
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 // __AIRSCHED_BOM_ORIGINDESTINATIONSETTYPES_HPP
00029
```

26.55 airsched/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 <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/SegmentPathPeriod.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

Functions

- template void [AIRSCHED::ReachableUniverse::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [AIRSCHED::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 // AirSched
00014 #include <airsched/bom/ReachableUniverse.hpp>
00015 #include <airsched/bom/SegmentPathPeriod.hpp>
00016
00017 namespace AIRSCHED {
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::serialisationImplementation() {
00049     std::ostringstream oStr;
00050     boost::archive::text_oarchive oa (oStr);
00051     oa << *this;
00052
00053     std::istringstream iStr;
00054     boost::archive::text_iarchive ia (iStr);
00055     ia >> *this;
00056 }
00057
00058 // //////////////////////////////////////
00059 template<class Archive>
00060 void ReachableUniverse::serialize (Archive& ioArchive,
00061                                   const unsigned int iFileVersion) {
00062     ioArchive & _key;
00063 }
00064
00065 // //////////////////////////////////////
00066 // Explicit template instantiation
00067 namespace ba = boost::archive;
00068 template
00069 void ReachableUniverse::serialize<ba::text_oarchive> (ba::text_oarchive&,
00070                                                       unsigned int);
00071 template
00072 void ReachableUniverse::serialize<ba::text_iarchive> (ba::text_iarchive&,
00073                                                       unsigned int);
00074 // //////////////////////////////////////
00075
00076 }
00077

```

26.57 airsched/bom/ReachableUniverse.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <airsched/bom/ReachableUniverseKey.hpp>
#include <airsched/bom/ReachableUniverseTypes.hpp>
#include <airsched/bom/SegmentPathPeriodTypes.hpp>
```

Classes

- class [AIRSCHED::ReachableUniverse](#)
Class representing the root of the schedule-related BOM tree.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.58 ReachableUniverse.hpp

```
00001 #ifndef __AIRSCHED_BOM_REACHABLEUNIVERSE_HPP
00002 #define __AIRSCHED_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 // AirSched
00013 #include <airsched/bom/ReachableUniverseKey.hpp>
00014 #include <airsched/bom/ReachableUniverseTypes.hpp>
00015 #include <airsched/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 AIRSCHED {
00029
00030
00031
```

```

00041 class ReachableUniverse : public stdair::BomAbstract {
00042     template <typename BOM> friend class stdair::FacBom;
00043     friend class stdair::FacBomManager;
00044     friend class SegmentPathGenerator;
00045     friend class boost::serialization::access;
00046
00047 public:
00048     // /////////////// Type definitions ///////////////
00049     typedef ReachableUniverseKey Key_T;
00050
00051 public:
00052     // /////////////// Getters ///////////////
00053     const Key_T& getKey() const {
00054         return _key;
00055     }
00056
00057     const stdair::AirportCode_T& getOrigin() const {
00058         return _key.getBoardingPoint();
00059     }
00060
00061     stdair::BomAbstract* const getParent() const {
00062         return _parent;
00063     }
00064
00065     const stdair::HolderMap_T& getHolderMap() const {
00066         return _holderMap;
00067     }
00068
00069     const SegmentPathPeriodListList_T& getSegmentPathPeriodListList() const {
00070         return _segmentPathPeriodListList;
00071     }
00072
00073 public:
00074     // /////////////// Display support methods ///////////////
00075     void toStream (std::ostream& ioOut) const {
00076         ioOut << toString();
00077     }
00078
00079     void fromStream (std::istream& ioIn) {
00080     }
00081
00082     std::string toString() const;
00083
00084     const std::string describeKey() const {
00085         return _key.toString();
00086     }
00087
00088 public:
00089     // /////////////// (Boost) Serialisation support methods ///////////////
00090     template<class Archive>
00091     void serialize (Archive& ar, const unsigned int iFileVersion);
00092
00093 private:
00094     void serialisationImplementation();
00095
00096 protected:
00097     // /////////////// Constructors and destructors ///////////////
00098     ReachableUniverse (const Key_T&);
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

```

```

00154     ~ReachableUniverse();
00155
00156 private:
00160     ReachableUniverse();
00161
00165     ReachableUniverse (const ReachableUniverse&);
00166
00167
00168 protected:
00169     // ////////// Attributes //////////
00173     Key_T _key;
00174
00178     stdair::BomAbstract* _parent;
00179
00183     stdair::HolderMap_T _holderMap;
00184
00190     SegmentPathPeriodListList_T _segmentPathPeriodListList;
00191 };
00192
00193 }
00194 #endif // __AIRSCHED_BOM_REACHABLEUNIVERSE_HPP
00195

```

26.59 airsched/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 <airsched/bom/ReachableUniverseKey.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

Functions

- template void [AIRSCHED::ReachableUniverseKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [AIRSCHED::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 // AirSched
00014 #include <airsched/bom/ReachableUniverseKey.hpp>
00015
00016 namespace AIRSCHED {
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::serialisationImplementation() {
00058     std::ostringstream oStr;
00059     boost::archive::text_oarchive oa (oStr);
00060     oa << *this;
00061
00062     std::istringstream iStr;
00063     boost::archive::text_iarchive ia (iStr);
00064     ia >> *this;

```

```

00065     }
00066
00067     // //////////////////////////////////////
00068     template<class Archive>
00069     void ReachableUniverseKey::serialize (Archive& ioArchive,
00070                                           const unsigned int iFileVersion) {
00071         ioArchive & _origin;
00072     }
00073
00074     // //////////////////////////////////////
00075     // Explicit template instantiation
00076     namespace ba = boost::archive;
00077     template
00078     void ReachableUniverseKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00079                                                             unsigned int);
00080     template
00081     void ReachableUniverseKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00082                                                             unsigned int);
00083     // //////////////////////////////////////
00084 }

```

26.61 airsched/bom/ReachableUniverseKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_basic_types.hpp>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [AIRSCHED::ReachableUniverseKey](#)
Structure representing the key of the schedule-related BOM tree root.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [AIRSCHED](#)

26.62 ReachableUniverseKey.hpp

```

00001 #ifndef __AIRSCHED_BOM_REACHABLEUNIVERSEKEY_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
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:
00099         // ////////// (Boost) Serialisation support methods //////////
00104         template<class Archive>
00105         void serialize (Archive& ar, const unsigned int iFileVersion);
00106
00107     private:
00112         void serialisationImplementation();
00113
00114     private:
00116         // ////////// Attributes //////////
00121         stdair::AirportCode_T _origin;
00122     };
00123
00124 }
00125
00126 #endif // __AIRSCHED_BOM_REACHABLEUNIVERSEKEY_HPP

```

26.63 airsched/bom/ReachableUniverseTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

Typedefs

- typedef std::list< ReachableUniverse * > [AIRSCHED::ReachableUniverseList_T](#)
- typedef std::map< const stdair::MapKey_T, ReachableUniverse * > [AIRSCHED::ReachableUniverseMap_T](#)

26.64 ReachableUniverseTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __AIRSCHED_BOM_REACHABLEUNIVERSETYPES_HPP
00003 #define __AIRSCHED_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 AIRSCHED {
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 // __AIRSCHED_BOM_REACHABLEUNIVERSETYPES_HPP
00029
```

26.65 airsched/bom/SegmentCabinStruct.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <stdair/bom/SegmentCabin.hpp>
#include <airsched/bom/SegmentCabinStruct.hpp>
```

Namespaces

- namespace AIRSCHED

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 // AIRSCHED
00010 #include <airsched/bom/SegmentCabinStruct.hpp>
00011
00012 namespace AIRSCHED {
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 airsched/bom/SegmentCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airsched/bom/FareFamilyStruct.hpp>
```

Classes

- struct AIRSCHED::SegmentCabinStruct

Namespaces

- namespace `stdair`
Forward declarations.
- namespace `AIRSCHEd`

Typedefs

- typedef `std::vector< SegmentCabinStruct >` `AIRSCHEd::SegmentCabinStructList_T`

26.68 SegmentCabinStruct.hpp

```

00001 #ifndef __AIRSCHEd_BOM_SEGMENTCABINSTRUCT_HPP
00002 #define __AIRSCHEd_BOM_SEGMENTCABINSTRUCT_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 // AirSched
00014 #include <airsched/bom/FareFamilyStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentCabin;
00019 }
00020
00021 namespace AIRSCHEd {
00022
00023     struct SegmentCabinStruct : public stdair::StructAbstract {
00024         // Attributes
00025         stdair::CabinCode_T _cabinCode;
00026         stdair::ClassList_String_T _classes;
00027         stdair::FamilyCode_T _itFamilyCode;
00028         FareFamilyStructList_T _fareFamilies;
00029
00030         void fill (stdair::SegmentCabin&) const;
00031
00032         const std::string describe() const;
00033
00034     };
00035
00036     typedef std::vector<SegmentCabinStruct> SegmentCabinStructList_T;
00037
00038 }
00039
00040 #endif // __AIRSCHEd_BOM_SEGMENTCABINSTRUCT_HPP

```

26.69 airsched/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 <airsched/bom/SegmentPathPeriod.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00021 #include <airsched/bom/SegmentPathPeriod.hpp>
00022
00023 namespace AIRSCHED {
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::serialisationImplementation() {
00060     std::ostringstream oStr;
00061     boost::archive::text_oarchive oa (oStr);
00062     oa << *this;
00063
00064     std::istringstream iStr;
00065     boost::archive::text_iarchive ia (iStr);
00066     ia >> *this;
00067 }
00068
00069 // //////////////////////////////////////
00070 template<class Archive>
00071 void SegmentPathPeriod::serialize (Archive& ioArchive,
00072                                   const unsigned int iFileVersion) {
00073     ioArchive & _key;
00074 }
00075
00076 // //////////////////////////////////////
00077 stdair::SegmentPeriod* SegmentPathPeriod::getLastSegmentPeriod () const {
00078     // Retrieve the last segment of the list
00079     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00080         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00081     stdair::SegmentPeriodList_T::const_reverse_iterator itLastSegment =
00082         lSegmentPeriodList.rbegin();
00083
00084     if (itLastSegment == lSegmentPeriodList.rend()) {
00085         return NULL;
00086     }

```

```

00087
00088     stdair::SegmentPeriod* oSegment_ptr = *itLastSegment;
00089     assert (oSegment_ptr != NULL);
00090
00091     return oSegment_ptr;
00092 }
00093
00094 // //////////////////////////////////////
00095 stdair::SegmentPeriod* SegmentPathPeriod::getFirstSegmentPeriod () const{
00096     // Retrieve the first segment of the list
00097     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00098         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00099     stdair::SegmentPeriodList_T::const_iterator itFirstSegment =
00100         lSegmentPeriodList.begin();
00101
00102     if (itFirstSegment == lSegmentPeriodList.end()) {
00103         return NULL;
00104     }
00105
00106     stdair::SegmentPeriod* oSegment_ptr = *itFirstSegment;
00107     assert (oSegment_ptr != NULL);
00108
00109     return oSegment_ptr;
00110 }
00111
00112 // //////////////////////////////////////
00113 const stdair::AirportCode_T& SegmentPathPeriod::getDestination () const {
00114     const stdair::SegmentPeriod* lLastSegment_ptr = getLastSegmentPeriod();
00115     assert (lLastSegment_ptr != NULL);
00116     return lLastSegment_ptr->getOffPoint();
00117 }
00118
00119 // //////////////////////////////////////
00120 bool SegmentPathPeriod::
00121 isAirlineFlown (const stdair::AirlineCode_T& iAirlineCode) const {
00122     bool oAirlineFlown = false;
00123
00124     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00125         stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00126     for (stdair::SegmentPeriodList_T::const_iterator itSegmentPeriod =
00127         lSegmentPeriodList.begin();
00128         itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
00129         const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
00130         assert (lSegmentPeriod_ptr != NULL);
00131
00132         const stdair::FlightPeriod& lFlightPeriod =
00133             stdair::BomManager::getParent<stdair::FlightPeriod>(*lSegmentPeriod_ptr);
00134
00135         const stdair::Inventory& lInventory =
00136             stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00137         const stdair::AirlineCode_T& lSegmentAirlineCode =
00138             lInventory.getAirlineCode ();
00139         if (lSegmentAirlineCode == iAirlineCode) {
00140             oAirlineFlown = true;
00141             break;
00142         }
00143     }
00144     return oAirlineFlown;
00145 }
00146
00147 // //////////////////////////////////////

```

```

00148 SegmentPathPeriodKey SegmentPathPeriod::
00149 connectWithAnotherSegment(const SegmentPathPeriod& iSingleSegmentPath) const {
00150     SegmentPathPeriodKey oSegmentPathPeriodKey;
00151
00152     // Retrieve the (only) segment period of the single segment path.
00153     const stdair::SegmentPeriod* lNextSegmentPeriod_ptr =
00154         iSingleSegmentPath.getFirstSegmentPeriod();
00155     assert (lNextSegmentPeriod_ptr != NULL);
00156
00157     // Retrieve the last segment period of the current segment path and check
00158     // if the combination of the last segment and the next segment that we
00159     // want to add to the current segment path will create a new segment
00160     // (i.e., the two segment period belongs to the same flight number).
00161     const stdair::SegmentPeriod* lLastSegmentPeriod_ptr = getLastSegmentPeriod ();
00162
00163     ;
00164     assert (lLastSegmentPeriod_ptr != NULL);
00165     const stdair::FlightPeriod& lLastFlightPeriod = stdair::BomManager::
00166         getParent<stdair::FlightPeriod> (*lLastSegmentPeriod_ptr);
00167     const stdair::Inventory& lLastInventory =
00168         stdair::BomManager::getParent<stdair::Inventory> (lLastFlightPeriod);
00169
00170     const stdair::FlightPeriod& lNextFlightPeriod = stdair::BomManager::
00171         getParent<stdair::FlightPeriod> (*lNextSegmentPeriod_ptr);
00172     const stdair::Inventory& lNextInventory =
00173         stdair::BomManager::getParent<stdair::Inventory> (lNextFlightPeriod);
00174
00175     if (lLastFlightPeriod.getFlightNumber() == lNextFlightPeriod.getFlightNumber()
00176         && lLastInventory.getAirlineCode() == lNextInventory.getAirlineCode()) {
00177         return oSegmentPathPeriodKey;
00178     }
00179
00180     // Check if the new segment period will create a circle.
00181     const stdair::AirportCode_T& lDestination =
00182         lNextSegmentPeriod_ptr->getOffPoint();
00183     if (checkCircle (lDestination) == true) {
00184         return oSegmentPathPeriodKey;
00185     }
00186
00187     // Check if a passenger can connect from the last segment of the
00188     // current segment path to the first segment of the to-be-added
00189     // segment path. If yes, build a new departure period for the new
00190     // segment path.
00191     DateOffsetList_T lBoardingDateOffsetList =
00192         getBoardingDateOffsetList ();
00193     const stdair::PeriodStruct& lCurrentDeparturePeriod = getDeparturePeriod();
00194     const stdair::PeriodStruct& lNextDeparturePeriod =
00195         iSingleSegmentPath.getDeparturePeriod();
00196     const stdair::Duration_T& lLastOffTime =
00197         lLastSegmentPeriod_ptr->getOffTime();
00198     const stdair::Duration_T& lNextBoardingTime =
00199         lNextSegmentPeriod_ptr->getBoardingTime();
00200     // If the next boarding time is later than the last off time, check if
00201     // the passengers will have enough time for the transfer. If the next
00202     // boarding time is earlier than the last off time, check if the passengers
00203     // can connect to a flight in the next day.
00204     if (lNextBoardingTime >= lLastOffTime) {
00205         const stdair::Duration_T lStopTime = lNextBoardingTime - lLastOffTime;
00206         if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
00207             return oSegmentPathPeriodKey;
00208         } else {
00209             // Calculate the date offset of the next segment compare to
00210             // the first one. In this case, this value is equal to the offset

```



```

00209         // of the off date of the last segment compare to the boarding date
00210         // of the first segment.
00211         const stdair::DateOffset_T& lLastBoardingDateOffset =
00212             lBoardingDateOffsetList.at (getNbOfSegments() - 1);
00213         const stdair::DateOffset_T lNextBoardingDateOffset =
00214             lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
00215             - lLastSegmentPeriod_ptr->getBoardingDateOffset();
00216         const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
00217             stdair::DateOffset_T (0) - lNextBoardingDateOffset;
00218
00219         // Compute the adjusted departure period of the next segment by
00220         // subtracting the origin one with the boarding date offset.
00221         const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
00222             lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);
00223
00224         // Build the intersection of the two periods.
00225         const stdair::PeriodStruct lNewDeparturePeriod =
00226             lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
00227         stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
00228             lNextSegmentPeriod_ptr->getElapsedTime();
00229         lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
00230         oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
00231         oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
00232     }
00233 } else {
00234     const stdair::Duration_T lStopTime =
00235         lNextBoardingTime - lLastOffTime + stdair::Duration_T (24, 0, 0);
00236     if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
00237         return oSegmentPathPeriodKey;
00238     } else {
00239         // Calculate the date offset of the next segment compare to
00240         // the first one.
00241         const stdair::DateOffset_T& lLastBoardingDateOffset =
00242             lBoardingDateOffsetList.at (getNbOfSegments() - 1);
00243         const stdair::DateOffset_T lNextBoardingDateOffset =
00244             lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
00245             - lLastSegmentPeriod_ptr->getBoardingDateOffset() +
00246             stdair::DateOffset_T (1);
00247         const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
00248             stdair::DateOffset_T (0) - lNextBoardingDateOffset;
00249
00250         // Compute the adjusted departure period of the next segment by
00251         // subtracting the origin one with the boarding date offset.
00252         const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
00253             lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);
00254
00255         // Build the intersection of the two periods.
00256         const stdair::PeriodStruct lNewDeparturePeriod =
00257             lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
00258         stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
00259             lNextSegmentPeriod_ptr->getElapsedTime();
00260         lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
00261         oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
00262         oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
00263     }
00264 }
00265
00266 const stdair::Duration_T& lBoardingTime = getBoardingTime();
00267 oSegmentPathPeriodKey.setBoardingTime (lBoardingTime);
00268 oSegmentPathPeriodKey.setBoardingDateOffsetList (lBoardingDateOffsetList);
00269
00270 return oSegmentPathPeriodKey;

```

```

00271     }
00272
00273     // //////////////////////////////////////
00274     bool SegmentPathPeriod::
00275     checkCircle (const stdair::AirlineCode_T& iDestination) const {
00276         const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00277             stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
00278         for (stdair::SegmentPeriodList_T::const_iterator itSegment =
00279             lSegmentPeriodList.begin();
00280             itSegment != lSegmentPeriodList.end(); ++itSegment) {
00281             const stdair::SegmentPeriod* lCurrentSegment_ptr = *itSegment;
00282             assert (lCurrentSegment_ptr != NULL);
00283             const stdair::AirlineCode_T& lCurrentBoardingPoint =
00284                 lCurrentSegment_ptr->getBoardingPoint();
00285             if (lCurrentBoardingPoint == iDestination) {
00286                 return true;
00287             }
00288         }
00289         return false;
00290     }
00291
00292     // //////////////////////////////////////
00293     bool SegmentPathPeriod::
00294     isDepartureDateValid (const stdair::Date_T& iDepartureDate) const {
00295         const stdair::PeriodStruct& lPeriod = getDeparturePeriod ();
00296
00297         // Check if the departure date is within the date range.
00298         const stdair::DatePeriod_T& lDeparturePeriod = lPeriod.getDateRange ();
00299         if (lDeparturePeriod.contains (iDepartureDate) == false) {
00300             return false;
00301         }
00302
00303         // Check if the departure date is valid within the DOW.
00304         // 0 = Sunday, 1 = Monday, etc.
00305         const short lDay = iDepartureDate.day_of_week ();
00306         const stdair::DoWStruct& lDoW = lPeriod.getDoW ();
00307         if (lDoW.getStandardDayOfWeek (lDay) == false) {
00308             return false;
00309         }
00310
00311         return true;
00312     }
00313
00314 }

```

26.71 airsched/bom/SegmentPathPeriod.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/bom/BomAbstract.hpp>

#include <airsched/bom/SegmentPathPeriodKey.hpp>

#include <airsched/bom/SegmentPathPeriodTypes.hpp>

```

Classes

- class `AIRSCHEd::SegmentPathPeriod`
Class representing a segment/path.

Namespaces

- namespace `boost`
Forward declarations.
- namespace `boost::serialization`
- namespace `stdair`
Forward declarations.
- namespace `AIRSCHEd`

26.72 SegmentPathPeriod.hpp

```

00001 #ifndef __AIRSCHEd_BOM_SEGMENTPATHPERIOD_HPP
00002 #define __AIRSCHEd_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 // AirSched
00013 #include <airsched/bom/SegmentPathPeriodKey.hpp>
00014 #include <airsched/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 AIRSCHEd {
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     public:
00037         // ////////////////////////////////// Type definitions //////////////////////////////////
00038         typedef SegmentPathPeriodKey Key_T;
00039
00040     };
00041
00042 }

```

```

00055 public:
00056     // //////////// Getters ////////////
00060     const Key_T& getKey() const {
00061         return _key;
00062     }
00063
00067     stdair::BomAbstract* const getParent() const {
00068         return _parent;
00069     }
00070
00072     const stdair::PeriodStruct& getDeparturePeriod() const {
00073         return _key.getPeriod();
00074     }
00075
00077     const DateOffsetList_T& getBoardingDateOffsetList () const {
00078         return _key.getBoardingDateOffsetList ();
00079     }
00080
00082     const stdair::NbOfSegments_T getNbOfSegments() const {
00083         return _key.getNbOfSegments ();
00084     }
00085
00087     const stdair::NbOfAirlines_T& getNbOfAirlines() const {
00088         return _key.getNbOfAirlines ();
00089     }
00090
00092     const stdair::Duration_T& getElapsedTime() const {
00093         return _key.getElapsedTime ();
00094     }
00095
00097     const stdair::Duration_T& getBoardingTime() const {
00098         return _key.getBoardingTime ();
00099     }
00100
00104     const stdair::HolderMap_T& getHolderMap() const {
00105         return _holderMap;
00106     }
00107
00113     stdair::SegmentPeriod* getLastSegmentPeriod() const;
00114
00120     stdair::SegmentPeriod* getFirstSegmentPeriod() const;
00121
00126     const stdair::AirportCode_T& getDestination() const;
00127
00128
00129 public:
00130     // //////////// Business methods ////////////
00148     Key_T connectWithAnotherSegment (const SegmentPathPeriod&) const;
00149
00155     bool checkCircle (const stdair::AirportCode_T&) const;
00156
00161     bool isAirlineFlown (const stdair::AirlineCode_T&) const;
00162
00167     bool isDepartureDateValid (const stdair::Date_T&) const;
00168
00169 public:
00170     // //////////// Display support methods ////////////
00176     void toStream (std::ostream& ioOut) const {
00177         ioOut << toString();
00178     }
00179
00185     void fromStream (std::istream& ioIn) {

```

```

00186     }
00187
00191     std::string toString() const;
00192
00196     const std::string describeKey() const {
00197         return _key.toString();
00198     }
00199
00200
00201 public:
00202     // /////////// (Boost) Serialisation support methods ///////////
00206     template<class Archive>
00207     void serialize (Archive& ar, const unsigned int iFileVersion);
00208
00209 private:
00214     void serialisationImplementation();
00215
00216
00217 protected:
00218     // /////////// Constructors and destructors ///////////
00222     SegmentPathPeriod (const Key_T&);
00223
00227     ~SegmentPathPeriod();
00228
00229 private:
00233     SegmentPathPeriod();
00234
00238     SegmentPathPeriod (const SegmentPathPeriod&);
00239
00240
00241 protected:
00242     // /////////// Attributes ///////////
00248     Key_T _key;
00249
00253     stdair::BomAbstract* _parent;
00254
00261     stdair::HolderMap_T _holderMap;
00262 };
00263
00264 }
00265 #endif // __AIRSCHED_BOM_SEGMENTPATHPERIOD_HPP
00266

```

26.73 airsched/bom/SegmentPathPeriodKey.cpp File Reference

```

#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 <airsched/bom/SegmentPathPeriodKey.hpp>
```

Namespaces

- namespace AIRSCHED

26.74 SegmentPathPeriodKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // Boost.Serialization
00007 #include <boost/archive/text_iarchive.hpp>
00008 #include <boost/archive/text_oarchive.hpp>
00009 #include <boost/serialization/access.hpp>
00010 // StdAir
00011 #include <stdair/basic/BasConst_General.hpp>
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/basic/BasConst_Period_BOM.hpp>
00014 #include <stdair/basic/BasConst_TravelSolution.hpp>
00015 // AirSched
00016 #include <airsched/bom/SegmentPathPeriodKey.hpp>
00017
00018 namespace AIRSCHED {
00019
00020 // //////////////////////////////////////
00021 SegmentPathPeriodKey::SegmentPathPeriodKey()
00022 : _period (stdair::BOOST_DEFAULT_DATE_PERIOD, stdair::DEFAULT_DOW_STRING),
00023   _boardingTime (stdair::NULL_BOOST_TIME_DURATION),
00024   _elapsed (stdair::NULL_BOOST_TIME_DURATION),
00025   _nbOfAirlines (stdair::DEFAULT_NBFAIRLINES) {
00026 }
00027
00028 // //////////////////////////////////////
00029 SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey& iSPPK)
00030 : _period (iSPPK._period),
00031   _boardingTime (iSPPK._boardingTime),
00032   _elapsed (iSPPK._elapsed),
00033   _boardingDateOffsetList (iSPPK._boardingDateOffsetList),
00034   _nbOfAirlines (iSPPK._nbOfAirlines) {
00035 }
00036
00037 // //////////////////////////////////////
00038 SegmentPathPeriodKey:
00039 SegmentPathPeriodKey (const stdair::PeriodStruct& iPeriod,
00040                       const stdair::Duration_T& iBoardingTime,
00041                       const stdair::Duration_T& iElapsedTime,
00042                       const DateOffsetList_T& iBoardingDateOffsetList,
00043                       const stdair::NbOfAirlines_T& iNbOfAirlines)
00044 : _period (iPeriod),
00045   _boardingTime (iBoardingTime),
00046   _elapsed (iElapsedTime),
00047   _boardingDateOffsetList (iBoardingDateOffsetList),
00048   _nbOfAirlines (iNbOfAirlines) {
00049 }
00050
00051 // //////////////////////////////////////
```

```

00052 SegmentPathPeriodKey::~SegmentPathPeriodKey() {
00053 }
00054
00055 // //////////////////////////////////////
00056 void SegmentPathPeriodKey::toStream (std::ostream& ioOut) const {
00057     ioOut << "SegmentPathPeriodKey: " << toString() << std::endl;
00058 }
00059
00060 // //////////////////////////////////////
00061 void SegmentPathPeriodKey::fromStream (std::istream& ioIn) {
00062 }
00063
00064 // //////////////////////////////////////
00065 const std::string SegmentPathPeriodKey::toString() const {
00066     std::ostringstream ostr;
00067     ostr << _period.describeShort() << ", "
00068         << _boardingTime << ", " << _elapsed << ", ";
00069
00070     for (DateOffsetList_T::const_iterator itOffset =
00071          _boardingDateOffsetList.begin();
00072          itOffset != _boardingDateOffsetList.end(); ++itOffset) {
00073         const stdair::DateOffset_T& lDateOffset = *itOffset;
00074         ostr << lDateOffset.days() << ", ";
00075     }
00076
00077     ostr << _nbOfAirlines ;
00078     return ostr.str();
00079 }
00080
00081 // //////////////////////////////////////
00082 void SegmentPathPeriodKey::serialisationImplementation() {
00083     std::ostringstream ostr;
00084     boost::archive::text_oarchive oa (ostr);
00085     oa << *this;
00086
00087     std::istringstream istr;
00088     boost::archive::text_iarchive ia (istr);
00089     ia >> *this;
00090 }
00091
00092 // //////////////////////////////////////
00093 template<class Archive>
00094 void SegmentPathPeriodKey::serialize (Archive& ioArchive,
00095                                       const unsigned int iFileVersion) {
00096     //ioArchive & _period & _boardingTime & _elapsed & _nbOfAirlines;
00097     std::string lBTStr = boost::posix_time::to_simple_string (_boardingTime);
00098     std::string lElapsedStr = boost::posix_time::to_simple_string (_elapsed);
00099     ioArchive & lBTStr & lElapsedStr & _nbOfAirlines;
00100 }
00101
00102 }
00103
00104 }
00105
00106 }

```

26.75 airsched/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 <airsched/bom/SegmentPathPeriodTypes.hpp>
```

Classes

- struct [AIRSCHEDED::SegmentPathPeriodKey](#)
Structure representing the key of a segment/path.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [AIRSCHEDED](#)

26.76 SegmentPathPeriodKey.hpp

```
00001 #ifndef __AIRSCHEDED_BOM_SEGMENTPATHPERIODKEY_HPP
00002 #define __AIRSCHEDED_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 // AirSched
00016 #include <airsched/bom/SegmentPathPeriodTypes.hpp>
00017
00018 namespace boost {
00019     namespace serialization {
00020         class access;
00021     }
00022 }
00023
00024 namespace AIRSCHEDED {
00025
00026     struct SegmentPathPeriodKey : public stdair::KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     public:
00031         SegmentPathPeriodKey (const stdair::PeriodStruct&,
00032                               const stdair::Duration_T& iBoardingTime,
00033                               const stdair::Duration_T& iElapsed,
00034                               const DateOffsetList_T&,
00035                               const stdair::NbOfAirlines_T&);
```



```

00046
00050     SegmentPathPeriodKey();
00051
00055     SegmentPathPeriodKey (const SegmentPathPeriodKey&);
00056
00060     ~SegmentPathPeriodKey();
00061
00062
00063 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 serialisationImplementation();
00185
00186
00187     private:
00188         // //////////// Attributes ////////////
00192         stdair::PeriodStruct _period;
00193
00197         stdair::Duration_T _boardingTime;
00198
00202         stdair::Duration_T _elapsed;
00203
00208         DateOffsetList_T _boardingDateOffsetList;
00209
00213         stdair::NbOfAirlines_T _nbOfAirlines;
00214     };
00215
00216 }
00217 #endif // __AIRSCHED_BOM_SEGMENTPATHPERIODKEY_HPP

```

26.77 airsched/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 [AIRSCHED](#)

Typedefs

- typedef std::list< SegmentPathPeriod * > AIRSCHED::SegmentPathPeriodList_
T
- typedef std::multimap< const stdair::MapKey_T, SegmentPathPeriod * > AIRSCHED::SegmentPathPeriodMultim
T
- typedef std::vector< const SegmentPathPeriod * > AIRSCHED::SegmentPathPeriodLightList_
T
- typedef std::vector< SegmentPathPeriodLightList_T > AIRSCHED::SegmentPathPeriodListList_
T
- typedef std::vector< stdair::DateOffset_T > AIRSCHED::DateOffsetList_T

26.78 SegmentPathPeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __AIRSCHED_BOM_SEGMENTPATHPERIODTYPES_HPP
00003 #define __AIRSCHED_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 AIRSCHED {
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 // __AIRSCHED_BOM_SEGMENTPATHPERIODTYPES_HPP
00039

```

26.79 airsched/bom/SegmentPeriodHelper.cpp File Reference

```

#include <cassert>

#include <stdair/basic/BasConst_General.hpp>

#include <stdair/bom/SegmentPeriod.hpp>

```

```
#include <airsched/bom/SegmentPeriodHelper.hpp>
```

Namespaces

- namespace AIRSCHED

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 // AIRSCHED
00010 #include <airsched/bom/SegmentPeriodHelper.hpp>
00011
00012 namespace AIRSCHED {
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 airsched/bom/SegmentPeriodHelper.hpp File Reference

```

#include <airsched/bom/LegStruct.hpp>
#include <airsched/bom/SegmentStruct.hpp>

```

Classes

- class [AIRSCHED::SegmentPeriodHelper](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.82 SegmentPeriodHelper.hpp

```

00001 #ifndef __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP
00002 #define __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP

```

```

00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // AIRSCHED
00008 #include <airsched/bom/LegStruct.hpp>
00009 #include <airsched/bom/SegmentStruct.hpp>
00010
00011 // Forward declarations
00012 namespace stdair {
00013     class SegmentPeriod;
00014 }
00015
00016 namespace AIRSCHED {
00019     class SegmentPeriodHelper {
00020     public:
00021         // ////////// Business Methods //////////
00024         static void fill (stdair::SegmentPeriod&, const SegmentStruct&);
00025
00028         static void fill (stdair::SegmentPeriod&, const LegStructList_T&);
00029     };
00030
00031 }
00032 #endif // __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP

```

26.83 airsched/bom/SegmentStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentDate.hpp>
#include <airsched/bom/SegmentStruct.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AIRSCHED
00010 #include <airsched/bom/SegmentStruct.hpp>
00011
00012 namespace AIRSCHED {
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 airsched/bom/SegmentStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airsched/bom/SegmentCabinStruct.hpp>

```

Classes

- struct [AIRSCHED::SegmentStruct](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

Typedefs

- typedef std::vector< SegmentStruct > [AIRSCHED::SegmentStructList_T](#)

26.86 SegmentStruct.hpp

```

00001 #ifndef __AIRSCHED_BOM_SEGMENTSTRUCT_HPP
00002 #define __AIRSCHED_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 // AirSched
00014 #include <airsched/bom/SegmentCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018     class SegmentDate;
00019 }
00020
00021 namespace AIRSCHED {
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 #endif // __AIRSCHED_BOM_SEGMENTSTRUCT_HPP

```

26.87 airsched/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/FlightPeriod.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp>
#include <airsched/bom/SegmentPeriodHelper.hpp>
#include <airsched/command/InventoryGenerator.hpp>
```

Namespaces

- namespace AIRSCHEd

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/FlightPeriod.hpp>
00015 #include <stdair/bom/SegmentPeriod.hpp>
00016 #include <stdair/factory/FacBomManager.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirSched
00019 #include <airsched/bom/FlightPeriodStruct.hpp>
00020 #include <airsched/bom/SegmentPeriodHelper.hpp>
00021 #include <airsched/command/InventoryGenerator.hpp>
00022
00023 namespace AIRSCHEd {
00024
00025 // //////////////////////////////////////
00026 void InventoryGenerator::
00027     createFlightPeriod (stdair::BomRoot& ioBomRoot,
00028                         const FlightPeriodStruct& iFlightPeriodStruct) {
00029
00030     const stdair::AirlineCode_T& lAirlineCode = iFlightPeriodStruct._airlineCode;
00031
00032     // Instantiate an inventory object (if not exist)
00033     // for the given key (airline code)
00034     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00035         getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00036     if (lInventory_ptr == NULL) {
00037         stdair::InventoryKey lKey (lAirlineCode);
00038     }
```

```

00039         lInventory_ptr =
00040             &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00041         stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00042         stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00043     }
00044     assert (lInventory_ptr != NULL);
00045
00046     // Create the flight-period key.
00047     const stdair::PeriodStruct lPeriod (iFlightPeriodStruct._dateRange,
00048                                         iFlightPeriodStruct._dow);
00049     const stdair::FlightPeriodKey
00050         lFlightPeriodKey (iFlightPeriodStruct._flightNumber, lPeriod);
00051
00052     // Check that the flight-period object is not already created.
00053     stdair::FlightPeriod* lFlightPeriod_ptr = stdair::BomManager::
00054         getObjectPtr<stdair::FlightPeriod> (*lInventory_ptr,
00055                                             lFlightPeriodKey.toString());
00056     if (lFlightPeriod_ptr != NULL) {
00057         throw stdair::ObjectCreationDuplicationException ("");
00058     }
00059     assert (lFlightPeriod_ptr == NULL);
00060
00061     // Instantiate a flight-period object with the given key.
00062     lFlightPeriod_ptr = &stdair::FacBom<stdair::FlightPeriod>::
00063         instance().create (lFlightPeriodKey);
00064     stdair::FacBomManager::addToListAndMap (*lInventory_ptr, *lFlightPeriod_ptr);
00065
00066     stdair::FacBomManager::linkWithParent (*lInventory_ptr, *lFlightPeriod_ptr);
00067
00068     // Create the segment-periods.
00069     createSegmentPeriods (*lFlightPeriod_ptr, iFlightPeriodStruct);
00070 }
00071
00072 // //////////////////////////////////////
00073 void InventoryGenerator::
00074 createSegmentPeriods (stdair::FlightPeriod& ioFlightPeriod,
00075                      const FlightPeriodStruct& iFlightPeriodStruct) {
00076
00077     // Iterate on the segment structures.
00078     const SegmentStructList_T& lSegmentList = iFlightPeriodStruct._segmentList;
00079     for (SegmentStructList_T::const_iterator itSegment = lSegmentList.begin();
00080          itSegment != lSegmentList.end(); ++itSegment) {
00081
00082         const SegmentStruct& lSegment = *itSegment;
00083
00084         // Set the segment-period primary key.
00085         const stdair::AirportCode_T& lBoardingPoint = lSegment._boardingPoint;
00086         const stdair::AirportCode_T& lOffPoint = lSegment._offPoint;
00087         const stdair::SegmentPeriodKey lSegmentPeriodKey (lBoardingPoint,
00088                                                         lOffPoint);
00089
00090         // Instantiate a segment-period with the key.
00091         stdair::SegmentPeriod& lSegmentPeriod = stdair::
00092             FacBom<stdair::SegmentPeriod>::instance().create (lSegmentPeriodKey);
00093         stdair::FacBomManager::addToListAndMap (ioFlightPeriod, lSegmentPeriod);
00094         stdair::FacBomManager::linkWithParent (ioFlightPeriod, lSegmentPeriod);
00095
00096         // Set the segment-period attributes.
00097         SegmentPeriodHelper::fill (lSegmentPeriod, lSegment);
00098         SegmentPeriodHelper::fill (lSegmentPeriod, iFlightPeriodStruct._legList);
00099     }
00100 }

```

```
00100
00101 }
```

26.89 airsched/command/InventoryGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHEDED_Types.hpp>
```

Classes

- class [AIRSCHEDED::InventoryGenerator](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHEDED](#)
- namespace [AIRSCHEDED::ScheduleParserHelper](#)

26.90 InventoryGenerator.hpp

```
00001 #ifndef __AIRSCHEDED_CMD_INVENTORYGENERATOR_HPP
00002 #define __AIRSCHEDED_CMD_INVENTORYGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirSched
00010 #include <airsched/AIRSCHEDED_Types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     class BomRoot;
00015     class FlightPeriod;
00016 }
00017
00018 namespace AIRSCHEDED {
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
00031     class InventoryGenerator : public stdair::CmdAbstract {
00032         // Only the following class may use methods of InventoryGenerator.
```

```

00033     // Indeed, as those methods build the BOM, it is not good to expose
00034     // them publicly.
00035     friend class FlightPeriodFileParser;
00036     friend class FFFlightPeriodFileParser;
00037     friend struct ScheduleParserHelper::doEndFlight;
00038     friend class ScheduleParser;
00039
00040 private:
00041     static void createFlightPeriod (stdair::BomRoot&,
00042                                     const FlightPeriodStruct&);
00043
00044     static void createSegmentPeriods (stdair::FlightPeriod&,
00045                                       const FlightPeriodStruct&);
00046
00047 };
00048
00049 #endif // __AIRSCHED_CMD_INVENTORYGENERATOR_HPP

```

26.91 airsched/command/OnDParser.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <airsched/command/OnDParserHelper.hpp>
#include <airsched/command/OnDParser.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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/bom/BomRoot.hpp>
00009 // AirSched
00010 #include <airsched/command/OnDParserHelper.hpp>
00011 #include <airsched/command/OnDParser.hpp>
00012
00013 namespace AIRSCHED {
00014
00015 // //////////////////////////////////////
00016 void OnDParser::generateOnDPeriods (const stdair::Filename_T& iFilename,
00017                                     stdair::BomRoot& ioBomRoot) {
00018
00019     // Check that the file path given as input corresponds to an actual file
00020     const bool doesExistAndIsReadable =

```

```

00021         stdair::BasFileMgr::doesExistAndIsReadable (iFilename);
00022
00023         if (doesExistAndIsReadable == false) {
00024             throw OnDInputFileNotFoundException ("The O&D file " + iFilename
00025                                                 + " does not exist or can not be "
00026                                                 + "read");
00027         }
00028
00029         // Initialise the O&D-Period file parser.
00030         OnDPeriodFileParser lOnDPeriodParser (iFilename, ioBomRoot);
00031
00032         // Parse the CSV-formatted O&D input file, and generate the
00033         // corresponding O&D-Period for the airlines.
00034         lOnDPeriodParser.generateOnDPeriods();
00035     }
00036
00037 }
```

26.93 airsched/command/OnDParser.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class [AIRSCHED::OnDParser](#)
Class wrapping the parser entry point.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.94 OnDParser.hpp

```

00001 #ifndef __AIRSCHED_CMD_ONDPARSER_HPP
00002 #define __AIRSCHED_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/command/CmdAbstract.hpp>
00012
00014 namespace stdair {
00015     class BomRoot;
```

```

00016 }
00017
00018 namespace AIRSCHED {
00019
00023     class OnDParser : public stdair::CmdAbstract {
00024     public:
00031         static void generateOnDPeriods (const stdair::Filename_T&,
00032                                         stdair::BomRoot&);
00033     };
00034
00035 }
00036 #endif // __AIRSCHED_CMD_ONDPARSER_HPP

```

26.95 airsched/command/OnDParserHelper.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airsched/command/OnDParserHelper.hpp>
#include <airsched/command/OnDPeriodGenerator.hpp>

```

Namespaces

- namespace [AIRSCHED](#)
- namespace [AIRSCHED::OnDParserHelper](#)

Functions

- [chset_t AIRSCHED::OnDParserHelper::alpha_cap_set_p](#) ("A-Z")
- [repeat_p_t AIRSCHED::OnDParserHelper::airport_p](#) (chset_t("0-9A-Z").derived(), 3, 3)
- [repeat_p_t AIRSCHED::OnDParserHelper::airline_code_p](#) (alpha_cap_set_p.derived(), 2, 3)
- [bounded4_p_t AIRSCHED::OnDParserHelper::year_p](#) (uint4_p.derived(), 2000u, 2099u)
- [bounded2_p_t AIRSCHED::OnDParserHelper::month_p](#) (uint2_p.derived(), 1u, 12u)
- [bounded2_p_t AIRSCHED::OnDParserHelper::day_p](#) (uint2_p.derived(), 1u, 31u)
- [bounded2_p_t AIRSCHED::OnDParserHelper::hours_p](#) (uint2_p.derived(), 0u, 23u)
- [bounded2_p_t AIRSCHED::OnDParserHelper::minutes_p](#) (uint2_p.derived(), 0u, 59u)
- [bounded2_p_t AIRSCHED::OnDParserHelper::seconds_p](#) (uint2_p.derived(), 0u, 59u)
- [chset_t AIRSCHED::OnDParserHelper::class_code_p](#) ("A-Z")

Variables

- uint2_p_t AIRSCHED::OnDParserHelper::uint2_p
- uint4_p_t AIRSCHED::OnDParserHelper::uint4_p
- uint1_4_p_t AIRSCHED::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 // AIRSCHED
00011 #include <airsched/command/OnDParserHelper.hpp>
00012 #include <airsched/command/OnDPeriodGenerator.hpp>
00013
00014 namespace AIRSCHED {
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         ond = ond_key
00275         >> +( ';' >> segment )
00276         >> ond_end[doEndOnD(self._bomRoot, self._onDPeriod)]
00277         ;
00278
00279         ond_end = boost::spirit::classic::ch_p(';')
00280         ;
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
00336 //
00337 // Entry class for the file parser
00338 //
00340
00341 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00342 OnDPeriodFileParser::OnDPeriodFileParser (const stdair::Filename_T& iFilename,
00343                                           stdair::BomRoot& ioBomRoot)
00344     : _filename (iFilename), _bomRoot (ioBomRoot) {
00345     init();
00346 }
00347
00348 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00349 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 }

```

```

00376
00377 // //////////////////////////////////////
00378 bool OnDPeriodFileParser::generateOnDPeriods () {
00379     bool oResult = false;
00380
00381     STDAIR_LOG_DEBUG ("Parsing O&D input file: " << _filename);
00382
00383     // Initialise the parser (grammar) with the helper/staging structure.
00384     OnDParserHelper::OnDParser lODParser (_bomRoot, _onDPeriod);
00385
00386     // Launch the parsing of the file and, thanks to the doEndOnD
00387     // call-back structure, filling the worldSchedule (Fares)
00388     boost::spirit::classic::parse_info<iterator_t> info =
00389         boost::spirit::classic::parse (_startIterator, _endIterator, lODParser,
00390                                         boost::spirit::classic::space_p);
00391
00392     // Retrieves whether or not the parsing was successful
00393     oResult = info.hit;
00394
00395     const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
00396     if (oResult == true) {
00397         STDAIR_LOG_DEBUG ("Parsing of O&D input file: " << _filename
00398                             << " succeeded: read " << info.length
00399                             << " characters. The input file has "
00400                             << hasBeenFullyReadStr
00401                             << "been fully read. Stop point: " << info.stop);
00402
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 airsched/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 <airsched/AIRSCHEDED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/OnDPeriodStruct.hpp>

```

Classes

- struct [AIRSCHEDED::OnDParserHelper::ParserSemanticAction](#)

- struct AIRSCHED::OnDParserHelper::storeOrigin
- struct AIRSCHED::OnDParserHelper::storeDestination
- struct AIRSCHED::OnDParserHelper::storeDateRangeStart
- struct AIRSCHED::OnDParserHelper::storeDateRangeEnd
- struct AIRSCHED::OnDParserHelper::storeStartRangeTime
- struct AIRSCHED::OnDParserHelper::storeEndRangeTime
- struct AIRSCHED::OnDParserHelper::storeAirlineCode
- struct AIRSCHED::OnDParserHelper::storeClassCode
- struct AIRSCHED::OnDParserHelper::doEndOnD
- struct AIRSCHED::OnDParserHelper::OnDParser
- struct AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >
- class AIRSCHED::OnDPeriodFileParser

Namespaces

- namespace stdair
Forward declarations.
- namespace AIRSCHED
- namespace AIRSCHED::OnDParserHelper

26.98 OnDParserHelper.hpp

```

00001 #ifndef __AIRSCHED_CMD_ONDPARSERHELPER_HPP
00002 #define __AIRSCHED_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 // AirSched
00015 #include <airsched/AIRSCHED_Types.hpp>
00016 #include <airsched/basic/BasParserTypes.hpp>
00017 #include <airsched/bom/OnDPeriodStruct.hpp>
00018
00019 // Forward declarations
00020 namespace stdair {
00021     class BomRoot;
00022 }
00023
00024 namespace AIRSCHED {
00025
00026     namespace OnDParserHelper {
00027
00028         // //////////////////////////////////////
00029         //
00030         // Semantic actions
00031         //

```

```

00032 // //////////////////////////////////////
00033 struct ParserSemanticAction {
00034     ParserSemanticAction (OnDPeriodStruct&);
00038     OnDPeriodStruct& _onDPeriod;
00039 };
00040
00042 struct storeOrigin : public ParserSemanticAction {
00044     storeOrigin (OnDPeriodStruct&);
00046     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00047 };
00048
00050 struct storeDestination : public ParserSemanticAction {
00052     storeDestination (OnDPeriodStruct&);
00054     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00055 };
00056
00058 struct storeDateRangeStart : public ParserSemanticAction {
00060     storeDateRangeStart (OnDPeriodStruct&);
00062     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00063 };
00064
00066 struct storeDateRangeEnd : public ParserSemanticAction {
00068     storeDateRangeEnd (OnDPeriodStruct&);
00070     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00071 };
00072
00074 struct storeStartRangeTime : public ParserSemanticAction {
00076     storeStartRangeTime (OnDPeriodStruct&);
00078     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00079 };
00080
00082 struct storeEndRangeTime : public ParserSemanticAction {
00084     storeEndRangeTime (OnDPeriodStruct&);
00086     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
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
00122 struct OnDParser :
00123     public boost::spirit::classic::grammar<OnDParser> {
00124
00129     OnDParser (stdair::BomRoot&, OnDPeriodStruct&);
00130
00131     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 // __AIRSCHED_CMD_ONDPARSERHELPER_HPP

```

26.99 airsched/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 <airsched/bom/OnDPeriodStruct.hpp>

```



```
#include <airsched/command/OnDPeriodGenerator.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00013 #include <airsched/bom/OnDPeriodStruct.hpp>
00014 #include <airsched/command/OnDPeriodGenerator.hpp>
00015
00016 namespace AIRSCHED {
00017
00018 // //////////////////////////////////////
00019 void OnDPeriodGenerator::
00020 createOnDPeriod (stdair::BomRoot& ioBomRoot,
00021                 const OnDPeriodStruct& iOnDPeriodStruct) {
00022 }
00023 }
```

26.101 airsched/command/OnDPeriodGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHED_Types.hpp>
```

Classes

- class [AIRSCHED::OnDPeriodGenerator](#)
Class handling the generation / instantiation of the O&D-Period BOM.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)
- namespace [AIRSCHED::OnDParserHelper](#)

26.102 OnDPeriodGenerator.hpp

```

00001 #ifndef __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP
00002 #define __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirSched
00010 #include <airsched/AIRSCHED_Types.hpp>
00011
00012 namespace stdair {
00013     class BomRoot;
00014 }
00015
00016 namespace AIRSCHED {
00017     struct OnDPeriodStruct_T;
00018     namespace OnDParserHelper {
00019         struct doEndOnD;
00020     }
00021
00022     class OnDPeriodGenerator : public stdair::CmdAbstract {
00023     friend class OnDPeriodFileParser;
00024     friend struct OnDParserHelper::doEndOnD;
00025     friend class OnDParser;
00026
00027     private:
00028         static void createOnDPeriod (stdair::BomRoot&, const OnDPeriodStruct&);
00029     };
00030 }
00031
00032 #endif // __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP

```

26.103 airsched/command/ScheduleParser.cpp File Reference

```

#include <cassert>
#include <string>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <airsched/command/SegmentPathGenerator.hpp>
#include <airsched/command/ScheduleParserHelper.hpp>
#include <airsched/command/ScheduleParser.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00011 #include <airsched/command/SegmentPathGenerator.hpp>
00012 #include <airsched/command/ScheduleParserHelper.hpp>
00013 #include <airsched/command/ScheduleParser.hpp>
00014
00015 namespace AIRSCHED {
00016
00017 // //////////////////////////////////////
00018 void ScheduleParser::generateInventories (const stdair::Filename_T& iFilename,
00019                                           stdair::BomRoot& ioBomRoot) {
00020
00021     // Check that the file path given as input corresponds to an actual file
00022     const bool doesExistAndIsReadable =
00023         stdair::BasFileMgr::doesExistAndIsReadable (iFilename);
00024
00025     if (doesExistAndIsReadable == false) {
00026         throw ScheduleInputFileNotFoundExpection ("The schedule file " + iFilename
00027                                                    + " does not exist or can not "
00028                                                    + "be read");
00029     }
00030
00031     // Initialise the Flight-Period file parser.
00032     FlightPeriodFileParser lFlightPeriodParser (ioBomRoot, iFilename);
00033
00034     // Parse the CSV-formatted schedule input file, and generate the
00035     // corresponding Inventories for the airlines.
00036     lFlightPeriodParser.generateInventories();
00037
00038     // Build the network from the schedule.
00039     SegmentPathGenerator::createSegmentPathNetwork (ioBomRoot);
00040 }
00041
00042 }

```

26.105 airsched/command/ScheduleParser.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [AIRSCHED::ScheduleParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.106 ScheduleParser.hpp

```

00001 #ifndef __AIRSCHED_CMD_SCHEDULEPARSER_HPP
00002 #define __AIRSCHED_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
00013 // Forward declarations.
00014 namespace stdair {
00015     class BomRoot;
00016 }
00017
00018 namespace AIRSCHED {
00019
00021     class ScheduleParser : public stdair::CmdAbstract {
00022     public:
00028         static void generateInventories (const stdair::Filename_T&,
00029                                         stdair::BomRoot&);
00030     };
00031 }
00032 #endif // __AIRSCHED_CMD_SCHEDULEPARSER_HPP

```

26.107 airsched/command/ScheduleParserHelper.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airsched/command/ScheduleParserHelper.hpp>
#include <airsched/command/InventoryGenerator.hpp>

```

Namespaces

- namespace [AIRSCHED](#)
- namespace [AIRSCHED::ScheduleParserHelper](#)

Functions

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

Variables

- int1_p_t AIRSCHED::ScheduleParserHelper::int1_p
- uint2_p_t AIRSCHED::ScheduleParserHelper::uint2_p
- uint4_p_t AIRSCHED::ScheduleParserHelper::uint4_p
- uint1_4_p_t AIRSCHED::ScheduleParserHelper::uint1_4_p
- int1_p_t AIRSCHED::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 // AIRSCHED
00011 #include <airsched/command/ScheduleParserHelper.hpp>
00012 #include <airsched/command/InventoryGenerator.hpp>
00013
00014 namespace AIRSCHED {

```

```

00015
00016 namespace ScheduleParserHelper {
00017
00018 // ////////////////////////////////////////////////////////////////////
00019 // Semantic actions
00020 // ////////////////////////////////////////////////////////////////////
00021
00022 ParserSemanticAction::
00023 ParserSemanticAction (FlightPeriodStruct& ioFlightPeriod)
00024 : _flightPeriod (ioFlightPeriod) {
00025 }
00026
00027 // ////////////////////////////////////////////////////////////////////
00028 storeAirlineCode::
00029 storeAirlineCode (FlightPeriodStruct& ioFlightPeriod)
00030 : ParserSemanticAction (ioFlightPeriod) {
00031 }
00032
00033 // ////////////////////////////////////////////////////////////////////
00034 void storeAirlineCode::operator() (iterator_t iStr,
00035                                     iterator_t iStrEnd) const {
00036     const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00037     _flightPeriod._airlineCode = lAirlineCode;
00038
00039     // As that's the beginning of a new flight, the list of legs
00040     // must be reset
00041     _flightPeriod._legList.clear();
00042 }
00043
00044 // ////////////////////////////////////////////////////////////////////
00045 storeFlightNumber::
00046 storeFlightNumber (FlightPeriodStruct& ioFlightPeriod)
00047 : ParserSemanticAction (ioFlightPeriod) {
00048 }
00049
00050 // ////////////////////////////////////////////////////////////////////
00051 void storeFlightNumber::operator() (unsigned int iNumber) const {
00052     _flightPeriod._flightNumber = iNumber;
00053 }
00054
00055 // ////////////////////////////////////////////////////////////////////
00056 storeDateRangeStart::
00057 storeDateRangeStart (FlightPeriodStruct& ioFlightPeriod)
00058 : ParserSemanticAction (ioFlightPeriod) {
00059 }
00060
00061 // ////////////////////////////////////////////////////////////////////
00062 void storeDateRangeStart::operator() (iterator_t iStr,
00063                                     iterator_t iStrEnd) const {
00064     _flightPeriod._dateRangeStart = _flightPeriod.getDate();
00065
00066     // Reset the number of seconds
00067     _flightPeriod._itSeconds = 0;
00068 }
00069
00070 // ////////////////////////////////////////////////////////////////////
00071 storeDateRangeEnd::
00072 storeDateRangeEnd (FlightPeriodStruct& ioFlightPeriod)
00073 : ParserSemanticAction (ioFlightPeriod) {
00074 }
00075
00076 // ////////////////////////////////////////////////////////////////////

```

```

00077 void storeDateRangeEnd::operator() (iterator_t iStr,
00078                                     iterator_t iStrEnd) const {
00079     // As a Boost date period (DatePeriod_T) defines the last day of
00080     // the period to be end-date - one day, we have to add one day to that
00081     // end date before.
00082     const stdair::DateOffset_T oneDay (1);
00083     _flightPeriod._dateRangeEnd = _flightPeriod.getDate() + oneDay;
00084
00085     // Transform the date pair (i.e., the date range) into a date period
00086     _flightPeriod._dateRange =
00087         stdair::DatePeriod_T (_flightPeriod._dateRangeStart,
00088                               _flightPeriod._dateRangeEnd);
00089
00090     // Reset the number of seconds
00091     _flightPeriod._itSeconds = 0;
00092 }
00093
00094 // //////////////////////////////////////
00095 storeDow::storeDow (FlightPeriodStruct& ioFlightPeriod)
00096 : ParserSemanticAction (ioFlightPeriod) {
00097 }
00098
00099 // //////////////////////////////////////
00100 void storeDow::operator() (iterator_t iStr, iterator_t iStrEnd) const {
00101     stdair::DOW_String_T lDow (iStr, iStrEnd);
00102     _flightPeriod._dow = lDow;
00103 }
00104
00105 // //////////////////////////////////////
00106 storeLegBoardingPoint::
00107 storeLegBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00108 : ParserSemanticAction (ioFlightPeriod) {
00109 }
00110
00111 // //////////////////////////////////////
00112 void storeLegBoardingPoint::operator() (iterator_t iStr,
00113                                         iterator_t iStrEnd) const {
00114     stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00115
00116     // If a leg has already been parsed, add it to the FlightPeriod
00117     if (_flightPeriod._legAlreadyDefined == true) {
00118         _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00119     } else {
00120         _flightPeriod._legAlreadyDefined = true;
00121     }
00122
00123     // Set the (new) boarding point
00124     _flightPeriod._itLeg._boardingPoint = lBoardingPoint;
00125
00126     // As that's the beginning of a new leg, the list of cabins
00127     // must be reset
00128     _flightPeriod._itLeg._cabinList.clear();
00129
00130     // Add the airport code if it is not already stored in the airport lists
00131     _flightPeriod.addAirport (lBoardingPoint);
00132 }
00133
00134 // //////////////////////////////////////
00135 storeLegOffPoint::
00136 storeLegOffPoint (FlightPeriodStruct& ioFlightPeriod)
00137 : ParserSemanticAction (ioFlightPeriod) {
00138 }

```

```

00139
00140 // //////////////////////////////////////
00141 void storeLegOffPoint::operator() (iterator_t iStr,
00142                                   iterator_t iStrEnd) const {
00143     stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00144     _flightPeriod._itLeg._offPoint = lOffPoint;
00145
00146     // Add the airport code if it is not already stored in the airport lists
00147     _flightPeriod.addAirport (lOffPoint);
00148 }
00149
00150 // //////////////////////////////////////
00151 storeBoardingTime::
00152 storeBoardingTime (FlightPeriodStruct& ioFlightPeriod)
00153     : ParserSemanticAction (ioFlightPeriod) {
00154 }
00155
00156 // //////////////////////////////////////
00157 void storeBoardingTime::operator() (iterator_t iStr,
00158                                   iterator_t iStrEnd) const {
00159     _flightPeriod._itLeg._boardingTime = _flightPeriod.getTime();
00160
00161     // Reset the number of seconds
00162     _flightPeriod._itSeconds = 0;
00163
00164     // Reset the date off-set
00165     _flightPeriod._dateOffset = 0;
00166 }
00167
00168 // //////////////////////////////////////
00169 storeOffTime::
00170 storeOffTime (FlightPeriodStruct& ioFlightPeriod)
00171     : ParserSemanticAction (ioFlightPeriod) {
00172 }
00173
00174 // //////////////////////////////////////
00175 void storeOffTime::operator() (iterator_t iStr,
00176                               iterator_t iStrEnd) const {
00177     _flightPeriod._itLeg._offTime = _flightPeriod.getTime();
00178
00179     // Reset the number of seconds
00180     _flightPeriod._itSeconds = 0;
00181
00182     // As the boarding date off set is optional, it can be set only
00183     // afterwards, based on the staging date off-set value
00184     // (_flightPeriod._dateOffset).
00185     const stdair::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00186     _flightPeriod._itLeg._boardingDateOffset = lDateOffset;
00187 }
00188
00189 // //////////////////////////////////////
00190 storeElapsedTime::
00191 storeElapsedTime (FlightPeriodStruct& ioFlightPeriod)
00192     : ParserSemanticAction (ioFlightPeriod) {
00193 }
00194
00195 // //////////////////////////////////////
00196 void storeElapsedTime::operator() (iterator_t iStr,
00197                                   iterator_t iStrEnd) const {
00198     _flightPeriod._itLeg._elapsed = _flightPeriod.getTime();
00199
00200     // Reset the number of seconds

```



```

00201     _flightPeriod._itSeconds = 0;
00202
00203     // As the boarding date off set is optional, it can be set only
00204     // afterwards, based on the staging date off-set value
00205     // (_flightPeriod._dateOffset).
00206     const stdair::DateOffset_T lDateOffset (_flightPeriod._dateOffset);
00207     _flightPeriod._itLeg._offDateOffset = lDateOffset;
00208 }
00209
00210 // //////////////////////////////////////
00211 storeLegCabinCode::
00212 storeLegCabinCode (FlightPeriodStruct& ioFlightPeriod)
00213 : ParserSemanticAction (ioFlightPeriod) {
00214 }
00215
00216 // //////////////////////////////////////
00217 void storeLegCabinCode::operator() (char iChar) const {
00218     _flightPeriod._itLegCabin._cabinCode = iChar;
00219     //std::cout << "Cabin code: " << iChar << std::endl;
00220 }
00221
00222 // //////////////////////////////////////
00223 storeCapacity::
00224 storeCapacity (FlightPeriodStruct& ioFlightPeriod)
00225 : ParserSemanticAction (ioFlightPeriod) {
00226 }
00227
00228 // //////////////////////////////////////
00229 void storeCapacity::operator() (double iReal) const {
00230     _flightPeriod._itLegCabin._capacity = iReal;
00231     //std::cout << "Capacity: " << iReal << std::endl;
00232
00233     // The capacity is the last (according to the arrival order
00234     // within the schedule input file) detail of the leg cabin. Hence,
00235     // when a capacity is parsed, it means that the full cabin
00236     // details have already been parsed as well: the cabin can
00237     // thus be added to the leg.
00238     _flightPeriod._itLeg._cabinList.push_back (_flightPeriod._itLegCabin);
00239 }
00240
00241 // //////////////////////////////////////
00242 storeSegmentSpecificity::
00243 storeSegmentSpecificity (FlightPeriodStruct& ioFlightPeriod)
00244 : ParserSemanticAction (ioFlightPeriod) {
00245 }
00246
00247 // //////////////////////////////////////
00248 void storeSegmentSpecificity::operator() (char iChar) const {
00249     if (iChar == '0') {
00250         _flightPeriod._areSegmentDefinitionsSpecific = false;
00251     } else {
00252         _flightPeriod._areSegmentDefinitionsSpecific = true;
00253     }
00254
00255     // Do a few sanity checks: the two lists should get exactly the same
00256     // content (in terms of airport codes). The only difference is that one
00257     // is a STL set, and the other a STL vector.
00258     assert (_flightPeriod._airportList.size()
00259         == _flightPeriod._airportOrderedList.size());
00260     assert (_flightPeriod._airportList.size() >= 2);
00261
00262     // Since all the legs have now been parsed, we get all the airports

```

```

00263         // and the segments may be built.
00264         _flightPeriod.buildSegments();
00265     }
00266
00267     // //////////////////////////////////////
00268     storeSegmentBoardingPoint::
00269     storeSegmentBoardingPoint (FlightPeriodStruct& ioFlightPeriod)
00270     : ParserSemanticAction (ioFlightPeriod) {
00271     }
00272
00273     // //////////////////////////////////////
00274     void storeSegmentBoardingPoint::operator() (iterator_t iStr,
00275                                                 iterator_t iStrEnd) const {
00276         stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00277         _flightPeriod._itSegment._boardingPoint = lBoardingPoint;
00278     }
00279
00280     // //////////////////////////////////////
00281     storeSegmentOffPoint::
00282     storeSegmentOffPoint (FlightPeriodStruct& ioFlightPeriod)
00283     : ParserSemanticAction (ioFlightPeriod) {
00284     }
00285
00286     // //////////////////////////////////////
00287     void storeSegmentOffPoint::operator() (iterator_t iStr,
00288                                           iterator_t iStrEnd) const {
00289         stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00290         _flightPeriod._itSegment._offPoint = lOffPoint;
00291     }
00292
00293     // //////////////////////////////////////
00294     storeSegmentCabinCode::
00295     storeSegmentCabinCode (FlightPeriodStruct& ioFlightPeriod)
00296     : ParserSemanticAction (ioFlightPeriod) {
00297     }
00298
00299     // //////////////////////////////////////
00300     void storeSegmentCabinCode::operator() (char iChar) const {
00301         _flightPeriod._itSegmentCabin._cabinCode = iChar;
00302     }
00303
00304     // //////////////////////////////////////
00305     storeClasses::
00306     storeClasses (FlightPeriodStruct& ioFlightPeriod)
00307     : ParserSemanticAction (ioFlightPeriod) {
00308     }
00309
00310     // //////////////////////////////////////
00311     void storeClasses::operator() (iterator_t iStr,
00312                                   iterator_t iStrEnd) const {
00313         std::string lClasses (iStr, iStrEnd);
00314         _flightPeriod._itSegmentCabin._classes = lClasses;
00315
00316         // The list of classes is the last (according to the arrival order
00317         // within the schedule input file) detail of the segment cabin. Hence,
00318         // when a list of classes is parsed, it means that the full segment
00319         // cabin details have already been parsed as well: the segment cabin
00320         // can thus be added to the segment.
00321         if (_flightPeriod._areSegmentDefinitionsSpecific == true) {
00322             _flightPeriod.addSegmentCabin (_flightPeriod._itSegment,
00323                                           _flightPeriod._itSegmentCabin);
00324         } else {

```

```

00325         _flightPeriod.addSegmentCabin (_flightPeriod._itSegmentCabin);
00326     }
00327 }
00328
00329 // //////////////////////////////////////
00330 storeFamilyCode::
00331 storeFamilyCode (FlightPeriodStruct& ioFlightPeriod)
00332 : ParserSemanticAction (ioFlightPeriod) {
00333 }
00334
00335 // //////////////////////////////////////
00336 void storeFamilyCode::operator() (int iCode) const {
00337     std::ostringstream ostr;
00338     ostr << iCode;
00339     _flightPeriod._itSegmentCabin._itFamilyCode = ostr.str();
00340 }
00341
00342 // //////////////////////////////////////
00343 storeFClasses::
00344 storeFClasses (FlightPeriodStruct& ioFlightPeriod)
00345 : ParserSemanticAction (ioFlightPeriod) {
00346 }
00347
00348 // //////////////////////////////////////
00349 void storeFClasses::operator() (iterator_t iStr,
00350                                iterator_t iStrEnd) const {
00351     std::string lClasses (iStr, iStrEnd);
00352     FareFamilyStruct lFareFamily(_flightPeriod._itSegmentCabin._itFamilyCode,
00353                                 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.addFareFamily (_flightPeriod._itSegment,
00362                                     _flightPeriod._itSegmentCabin,
00363                                     lFareFamily);
00364     } else {
00365         _flightPeriod.addFareFamily (_flightPeriod._itSegmentCabin,
00366                                     lFareFamily);
00367     }
00368 }
00369
00370 // //////////////////////////////////////
00371 doEndFlight::
00372 doEndFlight (stdair::BomRoot& ioBomRoot,
00373              FlightPeriodStruct& ioFlightPeriod)
00374 : ParserSemanticAction (ioFlightPeriod),
00375   _bomRoot (ioBomRoot) {
00376 }
00377
00378 // //////////////////////////////////////
00379 // void doEndFlight::operator() (char iChar) const {
00380 void doEndFlight::operator() (iterator_t iStr,
00381                               iterator_t iStrEnd) const {
00382
00383     assert (_flightPeriod._legAlreadyDefined == true);
00384     _flightPeriod._legList.push_back (_flightPeriod._itLeg);
00385
00386     // The lists of legs and cabins must be reset

```

```

00387     _flightPeriod._legAlreadyDefined = false;
00388     _flightPeriod._itLeg._cabinList.clear();
00389
00390     // DEBUG: Display the result
00391     STDAIR_LOG_DEBUG ("FlightPeriod: " << _flightPeriod.describe());
00392
00393     // Create the FlightPeriod BOM objects, and potentially the intermediary
00394     // objects (e.g., Inventory).
00395     InventoryGenerator::createFlightPeriod (_bomRoot, _flightPeriod);
00396 }
00397
00398
00399 // //////////////////////////////////////
00400 //
00401 // Utility Parsers
00402 //
00403 // //////////////////////////////////////
00405 int1_p_t int1_p;
00406
00408 uint2_p_t uint2_p;
00409
00411 uint4_p_t uint4_p;
00412
00414 uint1_4_p_t uint1_4_p;
00415
00417 repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3);
00418
00420 bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u);
00421
00423 bounded4_p_t year_p (uint4_p.derived(), 2000u, 2099u);
00424
00426 bounded2_p_t month_p (uint2_p.derived(), 1u, 12u);
00427
00429 bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00430
00432 repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7);
00433
00435 repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3);
00436
00438 bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u);
00439
00441 bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u);
00442
00444 bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u);
00445
00447 chset_t cabin_code_p ("A-Z");
00448
00450 int1_p_t family_code_p;
00451
00453 repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26);
00454
00455
00456 // //////////////////////////////////////
00457 // (Boost Spirit) Grammar Definition
00458 // //////////////////////////////////////
00459
00460 // //////////////////////////////////////
00461 FlightPeriodParser:
00462 FlightPeriodParser (stdair::BomRoot& ioBomRoot,
00463                     FlightPeriodStruct& ioFlightPeriod)
00464 : _bomRoot (ioBomRoot),
00465   _flightPeriod (ioFlightPeriod) {

```

```

00466     }
00467
00468     // //////////////////////////////////////
00469     template<typename ScannerT>
00470     FlightPeriodParser::definition<ScannerT>::
00471     definition (FlightPeriodParser const& self) {
00472
00473         flight_period_list = *( boost::spirit::classic::comment_p("//")
00474                                | boost::spirit::classic::comment_p("/*", "*/")
00475                                | flight_period )
00476         ;
00477
00478         flight_period = flight_key
00479         >> +( ';' >> leg )
00480         >> ';' >> segment_section
00481         >> flight_period_end[doEndFlight (self._bomRoot, self._flightPeriod)]
00482         ;
00483
00484         flight_period_end =
00485         boost::spirit::classic::ch_p(';')
00486         ;
00487
00488         flight_key = airline_code
00489         >> ';' >> flight_number
00490         >> ';' >> date[storeDateRangeStart (self._flightPeriod)]
00491         >> ';' >> date[storeDateRangeEnd (self._flightPeriod)]
00492         >> ';' >> dow[storeDow (self._flightPeriod)]
00493         ;
00494
00495         airline_code = boost::spirit::classic::
00496         lexeme_d[ (airline_code_p) [storeAirlineCode (self._flightPeriod)] ]
00497         ;
00498
00499         flight_number = boost::spirit::classic::
00500         lexeme_d[ (flight_number_p) [storeFlightNumber (self._flightPeriod)] ]
00501         ;
00502
00503         date = boost::spirit::classic::
00504         lexeme_d[ (year_p) [boost::spirit::classic::
00505             assign_a (self._flightPeriod._itYear) ]
00506             >> '-'
00507             >> (month_p) [boost::spirit::classic::
00508                 assign_a (self._flightPeriod._itMonth) ]
00509             >> '-'
00510             >> (day_p) [boost::spirit::classic::
00511                 assign_a (self._flightPeriod._itDay) ]
00512             ]
00513         ;
00514
00515         dow = boost::spirit::classic::lexeme_d[ dow_p ]
00516         ;
00517
00518         leg = leg_key >> ';' >> leg_details >> +( ';' >> leg_cabin_details )
00519         ;
00520
00521         leg_key =
00522         (airport_p) [storeLegBoardingPoint (self._flightPeriod)]
00523         >> ';'
00524         >> (airport_p) [storeLegOffPoint (self._flightPeriod)]
00525         ;
00526
00527         leg_details =

```

```

00528         time[storeBoardingTime(self._flightPeriod)]
00529     >> !(date_offset)
00530     >> ';'
00531     >> time[storeOffTime(self._flightPeriod)]
00532     >> !(date_offset)
00533     >> ';'
00534     >> time[storeElapsedTime(self._flightPeriod)]
00535     ;
00536
00537     time = boost::spirit::classic::
00538         lexeme_d[ (hours_p) [boost::spirit::classic::
00539             assign_a(self._flightPeriod._itHours)]
00540             >> ':'
00541             >> (minutes_p) [boost::spirit::classic::
00542                 assign_a(self._flightPeriod._itMinutes)]
00543             >> !(':'
00544                 >> (seconds_p) [boost::spirit::classic::
00545                     assign_a(self._flightPeriod._itSeconds)])
00546             ]
00547     ;
00548
00549     date_offset = boost::spirit::classic::ch_p('/')
00550     >> (int1_p) [boost::spirit::classic::
00551         assign_a(self._flightPeriod._dateOffset)]
00552     ;
00553
00554     leg_cabin_details = (cabin_code_p) [storeLegCabinCode(self._flightPeriod)]
00555     >> ';' >> (boost::spirit::classic::
00556         ureal_p) [storeCapacity(self._flightPeriod)]
00557     ;
00558
00559     segment_key =
00560         (airport_p) [storeSegmentBoardingPoint(self._flightPeriod)]
00561     >> ';'
00562     >> (airport_p) [storeSegmentOffPoint(self._flightPeriod)]
00563     ;
00564
00565     segment_section =
00566         generic_segment | specific_segment_list
00567     ;
00568
00569     generic_segment = boost::spirit::classic::
00570         ch_p('0') [storeSegmentSpecificity(self._flightPeriod)]
00571     >> +(';') >> segment_cabin_details)
00572     ;
00573
00574     specific_segment_list = boost::spirit::classic::
00575         ch_p('1') [storeSegmentSpecificity(self._flightPeriod)]
00576     >> +(';') >> segment_key >> full_segment_cabin_details)
00577     ;
00578
00579     full_segment_cabin_details =
00580         +(';') >> segment_cabin_details)
00581     ;
00582
00583     segment_cabin_details =
00584         (cabin_code_p) [storeSegmentCabinCode(self._flightPeriod)]
00585     >> ';' >> (class_code_list_p) [storeClasses(self._flightPeriod)]
00586     >> *(';') >> family_cabin_details)
00587     ;
00588
00589     family_cabin_details =

```

```

00590         (family_code_p)[storeFamilyCode(self._flightPeriod)]
00591     >> ' ';
00592     >> (class_code_list_p)[storeFCClasses(self._flightPeriod)]
00593     ;
00594
00595     // BOOST_SPIRIT_DEBUG_NODE (FlightPeriodParser);
00596     BOOST_SPIRIT_DEBUG_NODE (flight_period_list);
00597     BOOST_SPIRIT_DEBUG_NODE (flight_period);
00598     BOOST_SPIRIT_DEBUG_NODE (flight_period_end);
00599     BOOST_SPIRIT_DEBUG_NODE (flight_key);
00600     BOOST_SPIRIT_DEBUG_NODE (airline_code);
00601     BOOST_SPIRIT_DEBUG_NODE (flight_number);
00602     BOOST_SPIRIT_DEBUG_NODE (date);
00603     BOOST_SPIRIT_DEBUG_NODE (dow);
00604     BOOST_SPIRIT_DEBUG_NODE (leg);
00605     BOOST_SPIRIT_DEBUG_NODE (leg_key);
00606     BOOST_SPIRIT_DEBUG_NODE (leg_details);
00607     BOOST_SPIRIT_DEBUG_NODE (time);
00608     BOOST_SPIRIT_DEBUG_NODE (date_offset);
00609     BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
00610     BOOST_SPIRIT_DEBUG_NODE (segment_section);
00611     BOOST_SPIRIT_DEBUG_NODE (segment_key);
00612     BOOST_SPIRIT_DEBUG_NODE (generic_segment);
00613     BOOST_SPIRIT_DEBUG_NODE (specific_segment_list);
00614     BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
00615     BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
00616     BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
00617 }
00618
00619 // //////////////////////////////////////
00620 template<typename ScannerT>
00621 boost::spirit::classic::rule<ScannerT> const&
00622 FlightPeriodParser::definition<ScannerT>::start() const {
00623     return flight_period_list;
00624 }
00625
00626 }
00627
00628 //
00629 // Entry class for the file parser
00630 //
00631 // //////////////////////////////////////
00632 FlightPeriodFileParser::
00633 FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00634                         const stdair::Filename_T& iFilename)
00635 : _filename (iFilename), _bomRoot (ioBomRoot) {
00636     init();
00637 }
00638
00639 // //////////////////////////////////////
00640 void FlightPeriodFileParser::init() {
00641     // Check that the file exists and is readable
00642     const bool doesExistAndIsReadable =
00643         stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00644
00645     if (doesExistAndIsReadable == false) {
00646         STDAIR_LOG_ERROR ("The schedule file " << _filename
00647             << " does not exist or can not be read.");
00648
00649         throw ScheduleInputFileNotFoundException ("The schedule file " + _filename

```

```

00654                                     + " does not exist or can not be
read");
00655     }
00656
00657     // Open the file
00658     _startIterator = iterator_t (_filename);
00659
00660     // Check the filename exists and can be open
00661     if (!_startIterator) {
00662         STDAIR_LOG_ERROR ("The schedule file " << _filename << " can not be open."
00663                             << std::endl);
00664
00665         throw ScheduleInputFileNotFoundException ("The file " + _filename
00666                                                     + " does not exist or can not be
read");
00667     }
00668
00669     // Create an EOF iterator
00670     _endIterator = _startIterator.make_end();
00671 }
00672
00673 // //////////////////////////////////////
00674 bool FlightPeriodFileParser::generateInventories () {
00675     bool oResult = false;
00676
00677     STDAIR_LOG_DEBUG ("Parsing schedule input file: " << _filename);
00678
00679     // Initialise the parser (grammar) with the helper/staging structure.
00680     ScheduleParserHelper::FlightPeriodParser lFPParser (_bomRoot,
00681                                                         _flightPeriod);
00682
00683     // Launch the parsing of the file and, thanks to the doEndFlight
00684     // call-back structure, the building of the whole BomRoot BOM
00685     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00686     boost::spirit::classic::parse_info<iterator_t> info =
00687         boost::spirit::classic::parse (_startIterator, _endIterator, lFPParser,
00688                                         boost::spirit::classic::space_p);
00689
00690     // Retrieves whether or not the parsing was successful
00691     oResult = info.hit;
00692
00693     const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
00694     if (oResult == true) {
00695         STDAIR_LOG_DEBUG ("Parsing of schedule input file: " << _filename
00696                             << " succeeded: read " << info.length
00697                             << " characters. The input file has "
00698                             << hasBeenFullyReadStr
00699                             << "been fully read. Stop point: " << info.stop);
00700
00701     } else {
00702         // TODO: decide whether to throw an exception
00703         STDAIR_LOG_ERROR ("Parsing of schedule input file: " << _filename
00704                             << " failed: read " << info.length
00705                             << " characters. The input file has "
00706                             << hasBeenFullyReadStr
00707                             << "been fully read. Stop point: " << info.stop);
00708     }
00709
00710     return oResult;
00711 }
00712
00713 }

```


26.109 airsched/command/ScheduleParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHEDED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp>
```

Classes

- struct [AIRSCHEDED::ScheduleParserHelper::ParserSemanticAction](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeAirlineCode](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeFlightNumber](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeDateRangeStart](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeDateRangeEnd](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeDow](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeLegBoardingPoint](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeLegOffPoint](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeBoardingTime](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeOffTime](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeElapsedTime](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeLegCabinCode](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeCapacity](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeSegmentSpecificity](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeSegmentBoardingPoint](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeSegmentOffPoint](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeSegmentCabinCode](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeClasses](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeFamilyCode](#)
- struct [AIRSCHEDED::ScheduleParserHelper::storeFCClasses](#)
- struct [AIRSCHEDED::ScheduleParserHelper::doEndFlight](#)
- struct [AIRSCHEDED::ScheduleParserHelper::FlightPeriodParser](#)
- struct [AIRSCHEDED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >](#)
- class [AIRSCHEDED::FlightPeriodFileParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHEDED](#)
- namespace [AIRSCHEDED::ScheduleParserHelper](#)

26.110 ScheduleParserHelper.hpp

```

00001 #ifndef __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP
00002 #define __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // #define BOOST_SPIRIT_DEBUG
00010 // StdAir
00011 #include <stdair/command/CmdAbstract.hpp>
00012 // AirSched
00013 #include <airsched/AIRSCHED_Types.hpp>
00014 #include <airsched/basic/BasParserTypes.hpp>
00015 #include <airsched/bom/FlightPeriodStruct.hpp>
00016
00017 // Forward declarations
00018 namespace stdair {
00019     class BomRoot;
00020 }
00021
00022 namespace AIRSCHED {
00023
00024     namespace ScheduleParserHelper {
00025
00026         // //////////////////////////////////////
00027         // Semantic actions
00028         // //////////////////////////////////////
00029         struct ParserSemanticAction {
00030             ParserSemanticAction (FlightPeriodStruct&);
00031             FlightPeriodStruct& _flightPeriod;
00032         };
00033
00034         struct storeAirlineCode : public ParserSemanticAction {
00035             storeAirlineCode (FlightPeriodStruct&);
00036             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00037         };
00038
00039         struct storeFlightNumber : public ParserSemanticAction {
00040             storeFlightNumber (FlightPeriodStruct&);
00041             void operator() (unsigned int iNumber) const;
00042         };
00043
00044         struct storeDateRangeStart : public ParserSemanticAction {
00045             storeDateRangeStart (FlightPeriodStruct&);
00046             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00047         };
00048
00049         struct storeDateRangeEnd : public ParserSemanticAction {
00050             storeDateRangeEnd (FlightPeriodStruct&);
00051             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00052         };
00053
00054         struct storeDow : public ParserSemanticAction {
00055             storeDow (FlightPeriodStruct&);
00056             void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00057         };
00058
00059         struct storeLegBoardingPoint : public ParserSemanticAction {
00060             storeLegBoardingPoint (FlightPeriodStruct&);

```

```
00082     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00083 };
00084
00086 struct storeLegOffPoint : public ParserSemanticAction {
00088     storeLegOffPoint (FlightPeriodStruct&);
00090     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00091 };
00092
00094 struct storeBoardingTime : public ParserSemanticAction {
00096     storeBoardingTime (FlightPeriodStruct&);
00098     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00099 };
00100
00102 struct storeOffTime : public ParserSemanticAction {
00104     storeOffTime (FlightPeriodStruct&);
00106     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00107 };
00108
00110 struct storeElapsedTime : public ParserSemanticAction {
00112     storeElapsedTime (FlightPeriodStruct&);
00114     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00115 };
00116
00118 struct storeLegCabinCode : public ParserSemanticAction {
00120     storeLegCabinCode (FlightPeriodStruct&);
00122     void operator() (char iChar) const;
00123 };
00124
00126 struct storeCapacity : public ParserSemanticAction {
00128     storeCapacity (FlightPeriodStruct&);
00130     void operator() (double iReal) const;
00131 };
00132
00137 struct storeSegmentSpecificity : public ParserSemanticAction {
00139     storeSegmentSpecificity (FlightPeriodStruct&);
00141     void operator() (char iChar) const;
00142 };
00143
00145 struct storeSegmentBoardingPoint : public ParserSemanticAction {
00147     storeSegmentBoardingPoint (FlightPeriodStruct&);
00149     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00150 };
00151
00153 struct storeSegmentOffPoint : public ParserSemanticAction {
00155     storeSegmentOffPoint (FlightPeriodStruct&);
00157     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00158 };
00159
00161 struct storeSegmentCabinCode : public ParserSemanticAction {
00163     storeSegmentCabinCode (FlightPeriodStruct&);
00165     void operator() (char iChar) const;
00166 };
00167
00169 struct storeClasses : public ParserSemanticAction {
00171     storeClasses (FlightPeriodStruct&);
00173     void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00174 };
00175
00177 struct storeFamilyCode : public ParserSemanticAction {
00179     storeFamilyCode (FlightPeriodStruct&);
00181     void operator() (int iCode) const;
00182 };
```

```

00183
00185     struct storeFClasses : public ParserSemanticAction {
00187         storeFClasses (FlightPeriodStruct&);
00189         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00190     };
00191
00193     struct doEndFlight : public ParserSemanticAction {
00195         doEndFlight (stdair::BomRoot&, FlightPeriodStruct&);
00197         void operator() (iterator_t iStr, iterator_t iStrEnd) const;
00199         stdair::BomRoot& _bomRoot;
00200     };
00201
00202
00204     //
00205     // (Boost Spirit) Grammar Definition
00206     //
00208
00250     struct FlightPeriodParser :
00251         public boost::spirit::classic::grammar<FlightPeriodParser> {
00252
00253         FlightPeriodParser (stdair::BomRoot&, FlightPeriodStruct&);
00254
00255         template <typename ScannerT>
00256         struct definition {
00257             definition (FlightPeriodParser const& self);
00258
00259             // Instantiation of rules
00260             boost::spirit::classic::rule<ScannerT> flight_period_list, flight_period,
00261
00262                 flight_period_end, flight_key, airline_code, flight_number,
00263                 date, dow, time, date_offset,
00264                 leg, leg_key, leg_details, leg_cabin_details,
00265                 segment_section, segment_key, full_segment_cabin_details,
00266                 segment_cabin_details, full_family_cabin_details,
00267                 family_cabin_details, generic_segment, specific_segment_list;
00269             boost::spirit::classic::rule<ScannerT> const& start() const;
00270         };
00271
00272         // Parser Context
00273         stdair::BomRoot& _bomRoot;
00274         FlightPeriodStruct& _flightPeriod;
00275     };
00276
00277 }
00282
00283 //
00284 // Entry class for the file parser
00285 //
00287
00292 class FlightPeriodFileParser : public stdair::CmdAbstract {
00293 public:
00295     FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00296                             const stdair::Filename_T& iFilename);
00297
00299     bool generateInventories ();
00300
00301 private:
00303     void init();
00304
00305 private:
00306     // Attributes

```

```

00308     stdair::Filename_T _filename;
00309
00311     iterator_t _startIterator;
00312
00314     iterator_t _endIterator;
00315
00317     stdair::BomRoot& _bomRoot;
00318
00320     FlightPeriodStruct _flightPeriod;
00321 };
00322
00323 }
00324 #endif // __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP

```

26.111 airsched/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 <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/OriginDestinationSet.hpp>
#include <airsched/bom/SegmentPathPeriod.hpp>
#include <airsched/command/SegmentPathGenerator.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00017 #include <airsched/bom/ReachableUniverse.hpp>
00018 #include <airsched/bom/OriginDestinationSet.hpp>
00019 #include <airsched/bom/SegmentPathPeriod.hpp>
00020 #include <airsched/command/SegmentPathGenerator.hpp>
00021
00022 namespace AIRSCHED {
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                                                                 lDestination);
00125     if (lOriginDestinationSet_ptr == NULL) {
00126         OriginDestinationSetKey lKey (lDestination);
00127
00128         lOriginDestinationSet_ptr =
00129             &stdair::FacBom<OriginDestinationSet>::instance().create (lKey);
00130         stdair::FacBomManager::addToListAndMap (ioReachableUniverse,
00131
00132                                                                 *lOriginDestinationSet_ptr);

```

```

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
00170     addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod);
00171
00172     // Link the SegmentPathPeriod object with its parent, namely
00173     // OriginDestinationSet
00174     stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
00175                                       lSegmentPathPeriod);
00176     stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
00177                                           lSegmentPathPeriod);
00178
00179     // Link the SegmentPathPeriod and SegmentPeriod objects. Note that
00180     // the SegmentPeriod object has already a parent, namely FlightPeriod.
00181     stdair::FacBomManager::addToList (lSegmentPathPeriod,
00182                                       ioSegmentPeriod);
00183 }
00184
00185 // //////////////////////////////////////
00186 void SegmentPathGenerator::
00187 addSegmentPathPeriod (ReachableUniverse& ioReachableUniverse,
00188                     const SegmentPathPeriod& iSegmentPathPeriod) {
00189
00190     const stdair::NbOfSegments_T& lNbOfSegments =
00191         iSegmentPathPeriod.getNbOfSegments();
00192
00193     assert (lNbOfSegments > 0
00194           && lNbOfSegments <= stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND);
00195
00196     // If needed, initialise the list of lists with empty fixed-length
00197     // segment path period lists.
00198

```



```

00199     SegmentPathPeriodListList_T& lSegmentPathPeriodListList =
00200         ioReachableUniverse._segmentPathPeriodListList;
00201     while (lSegmentPathPeriodListList.size() < lNbOfSegments) {
00202         SegmentPathPeriodLightList_T lSegmentPathPeriodList;
00203         lSegmentPathPeriodListList.push_back (lSegmentPathPeriodList);
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) {

```

```

00266         return;
00267     }
00268
00269     // Retrieve the (i-1)-fixed-length segment path period list (i = number of
00270     // segments).
00271
00272     // Note that a STL vector starts at 0, whereas the number of segments
00273     // starts at 1. Hence, (i-1) for the length (in number of segments)
00274     // corresponds to [iNbOfSegments-2] for the STL vector.
00275
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
tination_iml =
00330         lSegmentPathPeriodListFromDestination_iml.at (0);
00331
00332     for (SegmentPathPeriodLightList_T::const_iterator
00333         itSegmentPathPeriodFromDestination_iml =
00334         lSingleSegmentPathPeriodLightListFromDestination_iml.begin();
00335         itSegmentPathPeriodFromDestination_iml
00336         != lSingleSegmentPathPeriodLightListFromDestination_iml.end();
00337         ++itSegmentPathPeriodFromDestination_iml) {
00338         const SegmentPathPeriod* lSingleSegmentPathPeriodFromDestination_iml_ptr=
00339
00340             *itSegmentPathPeriodFromDestination_iml;
00340         assert (lSingleSegmentPathPeriodFromDestination_iml_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_l_ptr =
00370             lSingleSegmentPathPeriodFromDestination_iml_ptr->getFirstSegmentPeriod(
00371 );
00372         assert (lSegmentPeriod_l_ptr != NULL);
00373
00374         // Calculate the number of airlines flown by the i-length
00375         // segment path period
00376         const stdair::FlightPeriod& lFlightPeriod = stdair::BomManager::
00377             getParent<stdair::FlightPeriod> (*lSegmentPeriod_l_ptr);
00378         const stdair::Inventory& lInventory =
00379             stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00380         const stdair::AirlineCode_T& lAirlineCode_l = lInventory.getAirlineCode();
00381
00382         stdair::NbOfAirlines_T lNbOfAirlines_i =
00383             lSegmentPathPeriod_iml_ptr->getNbOfAirlines();
00384         if (lSegmentPathPeriod_iml_ptr->isAirlineFlown(lAirlineCode_l) == false) {
00385
00386             ++lNbOfAirlines_i;
00387         }
00388     }

```

```

00384     }
00385     lSegmentPathPeriodKey_i.setNbOfAirlines (lNbOfAirlines_i);
00386
00387     // Create the new segment path and add it to the dedicated lists.
00388     OriginDestinationSet* lOriginDestinationSet_ptr = stdair::BomManager::
00389         getObjectPtr<OriginDestinationSet>(ioReachableUniverse,lDestination_i);
00390
00391     if (lOriginDestinationSet_ptr == NULL) {
00392         OriginDestinationSetKey lKey (lDestination_i);
00393         lOriginDestinationSet_ptr =
00394             &stdair::FacBom<OriginDestinationSet>::instance().create (lKey);
00395         stdair::FacBomManager::addToListAndMap (ioReachableUniverse,
00396             *lOriginDestinationSet_ptr);
00397         stdair::FacBomManager::linkWithParent (ioReachableUniverse,
00398             *lOriginDestinationSet_ptr);
00399     }
00400     assert (lOriginDestinationSet_ptr != NULL);
00401
00402     SegmentPathPeriod& lSegmentPathPeriod_i = stdair::
00403         FacBom<SegmentPathPeriod>::instance().create (lSegmentPathPeriodKey_i);
00404
00405     stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
00406         lSegmentPathPeriod_i);
00407     stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
00408         lSegmentPathPeriod_i);
00409
00410     // Clone the list of SegmentPeriod references of the given
00411     // SegmentPathPeriod object (passed as the second parameter).
00412     stdair::FacBomManager::
00413         cloneHolder<stdair::SegmentPeriod> (lSegmentPathPeriod_i,
00414             *lSegmentPathPeriod_i1_ptr);
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_1_ptr);
00423
00431     addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod_i);
00432     }
00433 }
00434 }
00435 }

```

26.113 airsched/command/SegmentPathGenerator.hpp File Reference

```

#include <vector>

#include <stdair/stdair_basic_types.hpp>

#include <stdair/command/CmdAbstract.hpp>

#include <airsched/AIRSCHEDED_Types.hpp>

```

Classes

- class `AIRSCHEd::SegmentPathGenerator`
Class handling the generation / instantiation of the network BOM.

Namespaces

- namespace `stdair`
Forward declarations.
- namespace `AIRSCHEd`

26.114 SegmentPathGenerator.hpp

```

00001 #ifndef __AIRSCHEd_CMD_SEGMENTPATHGENERATOR_HPP
00002 #define __AIRSCHEd_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 // AirSched
00013 #include <airsched/AIRSCHEd_Types.hpp>
00014
00016 namespace stdair {
00017     class BomRoot;
00018     class Inventory;
00019     class FlightPeriod;
00020     class SegmentPeriod;
00021 }
00022
00023 namespace AIRSCHEd {
00024
00026     class ReachableUniverse;
00027     class OriginDestinationSet;
00028     class SegmentPathPeriod;
00029
00030
00034     class SegmentPathGenerator : public stdair::CmdAbstract {
00035     public:
00039         static void createSegmentPathNetwork (const stdair::BomRoot&);
00040
00041     private:
00046         static void createSinglePaths (const stdair::Inventory&);
00047         static void createSinglePaths (const stdair::FlightPeriod&);
00048
00053         static void createSinglePath (stdair::SegmentPeriod&);
00054         static void createSinglePath (ReachableUniverse&, stdair::SegmentPeriod&);
00055
00059         static void buildSegmentPathNetwork (const stdair::BomRoot&,
00060                                             const stdair::NbOfSegments_T&);
00061         static void buildSegmentPathNetwork (ReachableUniverse&,
00062                                             const stdair::NbOfSegments_T&);
00063

```

```

00067     static void addSegmentPathPeriod (ReachableUniverse&,
00068                                       const SegmentPathPeriod&);
00069 };
00070
00071 }
00072 #endif // __AIRSCHED_CMD_SEGMENTPATHGENERATOR_HPP

```

26.115 airsched/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 <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/OriginDestinationSet.hpp>
#include <airsched/bom/SegmentPathPeriod.hpp>
#include <airsched/command/SegmentPathProvider.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00019 #include <airsched/bom/ReachableUniverse.hpp>
00020 #include <airsched/bom/OriginDestinationSet.hpp>
00021 #include <airsched/bom/SegmentPathPeriod.hpp>
00022 #include <airsched/command/SegmentPathProvider.hpp>
00023
00024 namespace AIRSCHED {
00025
00026 // //////////////////////////////////////
00027 void SegmentPathProvider::
00028 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00029                      const stdair::BomRoot& iBomRoot,
00030                      const stdair::BookingRequestStruct& iBookingRequest) {
00031     // Retrieve the reachable universe object corresponding to the
00032     // origin of the booking request.
00033     const stdair::AirportCode_T& lOrigin = iBookingRequest.getOrigin ();
00034     const ReachableUniverse* lReachableUniverse_ptr =
00035         stdair::BomManager::getObjectPtr<ReachableUniverse> (iBomRoot, lOrigin);
00036     if (lReachableUniverse_ptr != NULL) {
00037         buildSegmentPathList (ioTravelSolutionList, *lReachableUniverse_ptr,
00038                             iBookingRequest);
00039     }
00040 }
00041
00042 // //////////////////////////////////////
00043 void SegmentPathProvider::
00044 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00045                      const ReachableUniverse& iReachableUniverse,
00046                      const stdair::BookingRequestStruct& iBookingRequest) {
00047     // Retrieve the origin-destination set objet corresponding to the
00048     // destination of the booking request.
00049     const stdair::AirportCode_T& lDestination = iBookingRequest.getDestination();
00050
00051     const OriginDestinationSet* lOriginDestinationSet_ptr =
00052         stdair::BomManager::getObjectPtr<OriginDestinationSet> (iReachableUniverse,
00053                                                                 lDestination);
00054     if (lOriginDestinationSet_ptr != NULL) {
00055         buildSegmentPathList (ioTravelSolutionList, *lOriginDestinationSet_ptr,
00056                             iBookingRequest);
00057     }
00058 }
00059 // //////////////////////////////////////
00060 void SegmentPathProvider::
00061 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00062                      const OriginDestinationSet& iOriginDestinationSet,
00063                      const stdair::BookingRequestStruct& iBookingRequest) {
00064     // Retrieve the departure date of the booking request.
00065     const stdair::Date_T& lPreferedDepartureDate =
00066         iBookingRequest.getPreferedDepartureDate ();
00067
00068     // Browse the list of segment path periods and find those which content
00069     // the prefered departure date.
00070     const SegmentPathPeriodList_T& lSegmentPathPeriodList =
00071         stdair::BomManager::getList<SegmentPathPeriod> (iOriginDestinationSet);

```

```

00072     for (SegmentPathPeriodList_T::const_iterator itSegmentPath =
00073           lSegmentPathPeriodList.begin ();
00074           itSegmentPath != lSegmentPathPeriodList.end (); ++itSegmentPath) {
00075         const SegmentPathPeriod* lCurrentSegmentPath_ptr = *itSegmentPath;
00076         assert (lCurrentSegmentPath_ptr != NULL);
00077         if (lCurrentSegmentPath_ptr->isDepartureDateValid(lPreferedDepartureDate)) {

00078             buildSegmentPathList (ioTravelSolutionList, *lCurrentSegmentPath_ptr,
00079                                   iBookingRequest);
00080         }
00081     }
00082 }
00083
00084 // //////////////////////////////////////
00085 void SegmentPathProvider::
00086 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00087                       const SegmentPathPeriod& iSegmentPathPeriod,
00088                       const stdair::BookingRequestStruct& iBookingRequest) {
00089     // Create a new travel solution.
00090     stdair::TravelSolutionStruct lTravelSolution;
00091
00092     // Browse the list of segments and retrieve the necessary informations
00093     // for identifying the corresponding segment-date.
00094     const stdair::Date_T& lPreferedDepartureDate =
00095         iBookingRequest.getPreferedDepartureDate ();
00096     const stdair::SegmentPeriodList_T& lSegmentPeriodList =
00097         stdair::BomManager::getList<stdair::SegmentPeriod> (iSegmentPathPeriod);
00098     const DateOffsetList_T& lBoardingDateOffsetList =
00099         iSegmentPathPeriod.getBoardingDateOffsetList ();
00100     assert (lSegmentPeriodList.size() == lBoardingDateOffsetList.size());
00101     DateOffsetList_T::const_iterator itOffset = lBoardingDateOffsetList.begin();
00102     for (stdair::SegmentPeriodList_T::const_iterator itSegment =
00103           lSegmentPeriodList.begin();
00104           itSegment != lSegmentPeriodList.end(); ++itSegment) {
00105         const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegment;
00106         assert (lSegmentPeriod_ptr != NULL);
00107         const stdair::DateOffset_T& lBoardingDateOffset = *itOffset;
00108
00109         // Find the corresponding segment-date within the segment period.
00110         const stdair::DateOffset_T& lSegmentBoardingDateOffset =
00111             lSegmentPeriod_ptr->getBoardingDateOffset ();
00112         const stdair::Date_T& lReferenceFlightDate = lPreferedDepartureDate
00113             + lBoardingDateOffset - lSegmentBoardingDateOffset;
00114
00115         // Build the whole segment-date key string.
00116         const stdair::FlightPeriod& lFlightPeriod =
00117             stdair::BomManager::getParent<stdair::FlightPeriod>(*lSegmentPeriod_ptr);

00118         const stdair::Inventory& lInventory =
00119             stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
00120         const stdair::Duration_T lBoardingTime = lSegmentPeriod_ptr->getBoardingTim
00121 e();
00122         std::ostringstream oStr;
00123         oStr << lInventory.getAirlineCode()
00124             << stdair::DEFAULT_KEY_FLD_DELIMITER
00125             << lFlightPeriod.getFlightNumber()
00126             << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00127             << boost::gregorian::to_simple_string (lReferenceFlightDate)
00128             << stdair::DEFAULT_KEY_FLD_DELIMITER
00129             << lSegmentPeriod_ptr->getBoardingPoint ()
00130             << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00131             << lSegmentPeriod_ptr->getOffPoint ()

```



```

00131         << stdair::DEFAULT_KEY_FLD_DELIMITER
00132         << lBoardingTime;
00133
00134         lTravelSolution.addSegment (oStr.str());
00135
00136         ++itOffset;
00137     }
00138     ioTravelSolutionList.push_back (lTravelSolution);
00139 }
00140
00141 }
```

26.117 airsched/command/SegmentPathProvider.hpp File Reference

```

#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class [AIRSCHED::SegmentPathProvider](#)
Class building the travel solutions from airline schedules.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.118 SegmentPathProvider.hpp

```

00001 #ifndef __AIRSCHED_COM_CMD_SEGMENTPATHPROVIDER_HPP
00002 #define __AIRSCHED_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
00012 namespace stdair {
00013     class BomRoot;
00014     struct BookingRequestStruct;
00015 }
00016
00017 namespace AIRSCHED {
00018
00020     class ReachableUniverse;
00021     class OriginDestinationSet;
00022     class SegmentPathPeriod;
00023
00027     class SegmentPathProvider : public stdair::CmdAbstract {
```

```

00028     friend class AIRSCHED_Service;
00029
00030 private:
00031     // ////////////////////////////////// Business Methods //////////////////////////////////
00042     static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00043                                     const stdair::BomRoot&,
00044                                     const stdair::BookingRequestStruct&);
00045
00056     static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00057                                     const ReachableUniverse&,
00058                                     const stdair::BookingRequestStruct&);
00059
00070     static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00071                                     const OriginDestinationSet&,
00072                                     const stdair::BookingRequestStruct&);
00073
00084     static void buildSegmentPathList (stdair::TravelSolutionList_T&,
00085                                     const SegmentPathPeriod&,
00086                                     const stdair::BookingRequestStruct&);
00087 };
00088
00089 }
00090 #endif // __AIRSCHED_COM_CMD_SEGMENTPATHPROVIDER_HPP

```

26.119 airsched/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 <airsched/command/Simulator.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AIRSCHED
00014 #include <airched/command/Simulator.hpp>
00015
00016 namespace AIRSCHED {
00017
00018 // //////////////////////////////////////
00019 void Simulator::simulate (stdair::BomRoot& ioBomRoot) {
00020
00021     // Delegate to the dedicated StdAir utility class
00022     // std::ostream 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 airtched/command/Simulator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class [AIRSCHED::Simulator](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [AIRSCHED](#)

26.122 Simulator.hpp

```

00001 #ifndef __AIRSCHED_COM_CMD_SIMULATOR_HPP
00002 #define __AIRSCHED_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 AIRSCHED {
00016
00018     class Simulator : public stdair::CmdAbstract {
00019     public:
00020
00021         // //////////// Business Methods ////////////
00022         static void simulate (stdair::BomRoot&);
00023     };
00024
00025 }
00026
00027 }
00028 #endif // __AIRSCHED_COM_CMD_SIMULATOR_HPP

```

26.123 airsched/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 <airsched/command/TravelSolutionParser.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

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 // AirSched
00015 #include <airsched/command/TravelSolutionParser.hpp>

```

```

00016
00017 namespace AIRSCHED {
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 hasAllPArms = 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             hasAllPArms = 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
00134         //
00135         ++i;
00136
00137     } else {
00138         hasAllPParams = 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 airsched/command/TravelSolutionParser.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [AIRSCHEDED::TravelSolutionParser](#)
Class filling the TravelSolutionHolder structure (representing a list of classes/travelSolutions) from a given input file.

Namespaces

- namespace [AIRSCHEDED](#)

26.126 TravelSolutionParser.hpp

```

00001 #ifndef __AIRSCHEDED_CMD_TRAVELSOLUTIONPARSER_HPP
00002 #define __AIRSCHEDED_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 AIRSCHEDED {
00014

```

```

00019 class TravelSolutionParser : public stdair::CmdAbstract {
00020 public:
00028     static bool parseInputFileAndBuildBom (const stdair::Filename_T&);
00029 };
00030 }
00031 #endif // __AIRSCHED_CMD_TRAVELSOLUTIONPARSER_HPP

    */
#ifdef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_PATHS_HPP__

#define PACKAGE "airsched"
#define PACKAGE_NAME "AIRSCHED"
#define PACKAGE_VERSION "0.1.2"
#define PREFIXDIR "/usr"
#define EXEC_PREFIX "/usr"
#define BINDIR "/usr/bin"
#define LIBDIR "/usr/lib"
#define LIBEXECDIR "/usr/libexec"
#define SBINDIR "/usr/sbin"
#define SYSCONFDIR "/usr/etc"
#define INCLUDEDIR "/usr/include"
#define DATAROOTDIR "/usr/share"
#define DATADIR "/usr/share"
#define DOCDIR "/usr/share/doc/airsched-0.1.2"
#define MANDIR "/usr/share/man"
#define INFODIR "/usr/share/info"
#define HTMLDIR "/usr/share/doc/airsched-0.1.2/html"
#define PDFDIR "/usr/share/doc/airsched-0.1.2/html"
#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

#endif // __AIRSCHED_PATHS_HPP__

/*!

    */
#ifdef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_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 // __AIRSCHED_PATHS_HPP__

/*!

```


26.127 airsched-paths.hpp

```

00001
00005 #ifndef __AIRSCHED_PATHS_HPP__
00006 #define __AIRSCHED_PATHS_HPP__
00007
00008 #define PACKAGE "airsched"
00009 #define PACKAGE_NAME "AIRSCHED"
00010 #define PACKAGE_VERSION "0.1.2"
00011 #define PREFIXDIR "/usr"
00012 #define EXEC_PREFIX "/usr"
00013 #define BINDIR "/usr/bin"
00014 #define LIBDIR "/usr/lib"
00015 #define LIBEXECDIR "/usr/libexec"
00016 #define SBINDIR "/usr/sbin"
00017 #define SYSCONFDIR "/usr/etc"
00018 #define INCLUDEDIR "/usr/include"
00019 #define DATAROOTDIR "/usr/share"
00020 #define DATADIR "/usr/share"
00021 #define DOCDIR "/usr/share/doc/airsched-0.1.2"
00022 #define MANDIR "/usr/share/man"
00023 #define INFODIR "/usr/share/info"
00024 #define HTMLDIR "/usr/share/doc/airsched-0.1.2/html"
00025 #define PDFDIR "/usr/share/doc/airsched-0.1.2/html"
00026 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00027
00028 #endif // __AIRSCHED_PATHS_HPP__
00029

```

26.128 airsched/config/airsched-paths.hpp.in File Reference**26.129 airsched-paths.hpp.in**

```

00001
00005 #ifndef __AIRSCHED_PATHS_HPP__
00006 #define __AIRSCHED_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 // __AIRSCHED_PATHS_HPP__
00029

```

26.130 airsched/factory/FacAIRSCHEDServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airsched/factory/FacAIRSCHEDServiceContext.hpp>
#include <airsched/service/AIRSCHED_ServiceContext.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

26.131 FacAIRSCHEDServiceContext.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirSched
00009 #include <airsched/factory/FacAIRSCHEDServiceContext.hpp>
00010 #include <airsched/service/AIRSCHED_ServiceContext.hpp>
00011
00012 namespace AIRSCHED {
00013
00014     FacAIRSCHEDServiceContext* FacAIRSCHEDServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacAIRSCHEDServiceContext::~FacAIRSCHEDServiceContext () {
00018         _instance = NULL;
00019     }
00020
00021     // //////////////////////////////////////
00022     FacAIRSCHEDServiceContext& FacAIRSCHEDServiceContext::instance () {
00023
00024         if (_instance == NULL) {
00025             _instance = new FacAIRSCHEDServiceContext();
00026             assert (_instance != NULL);
00027
00028             stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029         }
00030         return *_instance;
00031     }
00032
00033     // //////////////////////////////////////
00034     AIRSCHED_ServiceContext& FacAIRSCHEDServiceContext::create () {
00035         AIRSCHED_ServiceContext* aServiceContext_ptr = NULL;
00036
00037         aServiceContext_ptr = new AIRSCHED_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 }

```

26.132 airsched/factory/FacAIRSCHEDServiceContext.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

```

Classes

- class [AIRSCHED::FacAIRSCHEDServiceContext](#)
Factory for the service context.

Namespaces

- namespace [AIRSCHED](#)

26.133 FacAIRSCHEDServiceContext.hpp

```

00001 #ifndef __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_HPP
00002 #define __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_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 AIRSCHED {
00012
00014     class AIRSCHED_ServiceContext;
00015
00019     class FacAIRSCHEDServiceContext : public stdair::FacServiceAbstract {
00020     public:
00021
00028         static FacAIRSCHEDServiceContext& instance();
00029
00036         ~FacAIRSCHEDServiceContext();
00037
00045         AIRSCHED_ServiceContext& create();
00046
00047
00048     protected:
00054         FacAIRSCHEDServiceContext() {}
00055
00056     private:
00060         static FacAIRSCHEDServiceContext* _instance;
00061
00062     };
00063 }

```

```
00064 #endif // __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_HPP
```

26.134 airsched/factory/FacServiceAbstract.cpp File Reference

```
#include <assert.h>
#include <airsched/service/ServiceAbstract.hpp>
#include <airsched/factory/FacServiceAbstract.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

26.135 FacServiceAbstract.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // TRAVEL-CCM
00007 #include <airsched/service/ServiceAbstract.hpp>
00008 #include <airsched/factory/FacServiceAbstract.hpp>
00009
00010 namespace AIRSCHED {
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 }
```

26.136 airsched/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class [AIRSCHED::FacServiceAbstract](#)

Namespaces

- namespace [AIRSCHED](#)

26.137 FacServiceAbstract.hpp

```

00001 #ifndef __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP
00002 #define __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRSCHED {
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 // __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP

```

26.138 airsched/service/AIRSCHED_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 <airsched/basic/BasConst_AIRSCHED_Service.hpp>
#include <airsched/factory/FacAIRSCHEDServiceContext.hpp>
#include <airsched/command/Simulator.hpp>
#include <airsched/command/ScheduleParser.hpp>
#include <airsched/command/OnDParser.hpp>
#include <airsched/command/SegmentPathProvider.hpp>
#include <airsched/command/InventoryGenerator.hpp>
#include <airsched/command/SegmentPathGenerator.hpp>
#include <airsched/service/AIRSCHED_ServiceContext.hpp>
#include <airsched/AIRSCHED_Service.hpp>

```

Namespaces

- namespace [AIRSCHED](#)

26.139 AIRSCHED_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 // AirSched
00017 #include <airsched/basic/BasConst_AIRSCHED_Service.hpp>
00018 #include <airsched/factory/FacAIRSCHEDServiceContext.hpp>
00019 #include <airsched/command/Simulator.hpp>
00020 #include <airsched/command/ScheduleParser.hpp>
00021 #include <airsched/command/OnDParser.hpp>
00022 #include <airsched/command/SegmentPathProvider.hpp>
00023 #include <airsched/command/InventoryGenerator.hpp>
00024 #include <airsched/command/SegmentPathGenerator.hpp>
00025 #include <airsched/service/AIRSCHED_ServiceContext.hpp>
00026 #include <airsched/AIRSCHED_Service.hpp>
00027
00028 namespace AIRSCHED {
00029
00030 // //////////////////////////////////////

```

```

00031 AIRSCHED_Service::AIRSCHED_Service() : _airschedServiceContext (NULL) {
00032     assert (false);
00033 }
00034
00035 // //////////////////////////////////////
00036 AIRSCHED_Service::AIRSCHED_Service (const AIRSCHED_Service& iService)
00037 : _airschedServiceContext (NULL) {
00038     assert (false);
00039 }
00040
00041 // //////////////////////////////////////
00042 AIRSCHED_Service::AIRSCHED_Service (const stdair::BasLogParams& iLogParams)
00043 : _airschedServiceContext (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 AirSched service context
00053     // \note AirSched 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     initAirschedService();
00059 }
00060
00061 // //////////////////////////////////////
00062 AIRSCHED_Service::AIRSCHED_Service (const stdair::BasLogParams& iLogParams,
00063                                     const stdair::BasDBParams& iDBParams)
00064 : _airschedServiceContext (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 AirSched service context
00074     // \note AirSched 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     initAirschedService();
00080 }
00081
00082 // //////////////////////////////////////
00083 AIRSCHED_Service::
00084 AIRSCHED_Service (stdair::STDAIR_ServicePtr_T ioSTDAIRServicePtr)
00085 : _airschedServiceContext (NULL) {
00086
00087     // Initialise the service context
00088     initServiceContext();
00089
00090     // Add the StdAir service context to the AirSched service context.
00091     // \note AirSched does not own the STDAIR service resources here.
00092     const bool doesNotOwnStdairService = false;

```

```

00093     addStdAirService (ioSTDAIRServicePtr, doesNotOwnStdairService);
00094
00095     // Initialise the context
00096     initAirschedService();
00097 }
00098
00099 // //////////////////////////////////////
00100 AIRSCHED_Service::~AIRSCHED_Service() {
00101     // Delete/Clean all the objects from memory
00102     finalise();
00103 }
00104
00105 // //////////////////////////////////////
00106 void AIRSCHED_Service::finalise() {
00107     assert (_airschedServiceContext != NULL);
00108     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00109     _airschedServiceContext->reset();
00110 }
00111
00112 // //////////////////////////////////////
00113 void AIRSCHED_Service::initServiceContext() {
00114     // Initialise the service context
00115     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00116         FacAIRSCHEDServiceContext::instance().create();
00117     _airschedServiceContext = &lAIRSCHED_ServiceContext;
00118 }
00119
00120 // //////////////////////////////////////
00121 void AIRSCHED_Service::
00122 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00123                  const bool iOwnStdairService) {
00124
00125     // Retrieve the AirSched service context
00126     assert (_airschedServiceContext != NULL);
00127     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00128         *_airschedServiceContext;
00129
00130     // Store the STDAIR service object within the (AirSched) service context
00131     lAIRSCHED_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00132                                                iOwnStdairService);
00133 }
00134
00135 // //////////////////////////////////////
00136 stdair::STDAIR_ServicePtr_T AIRSCHED_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 AIRSCHED_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 stdair::STDAIR_ServicePtr_T AIRSCHED_Service::
00158 initStdAirService (const stdair::BasLogParams& iLogParams,
00159                  const stdair::BasDBParams& iDBParams) {
00160
00161     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00162         boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00163
00164     return lSTDAIR_Service_ptr;
00165 }
00166
00167 // //////////////////////////////////////
00168 stdair::STDAIR_ServicePtr_T AIRSCHED_Service::
00169 initStdAirService (const stdair::BasLogParams& iLogParams,
00170                  const stdair::BasDBParams& iDBParams) {
00171
00172     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00173         boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00174
00175     return lSTDAIR_Service_ptr;
00176 }

```



```

00169
00170 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00171 void AIRSCHED_Service::initAirschedService() {
00172     // Do nothing at this stage. A sample BOM tree may be built by
00173     // calling the buildSampleBom() method
00174 }
00175
00176 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00177 void AIRSCHED_Service::
00178 parseAndLoad (const stdair::Filename_T& iScheduleInputFilename) {
00179
00180     // Retrieve the BOM root object.
00181     assert (_airschedServiceContext != NULL);
00182     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00183         *_airschedServiceContext;
00184     stdair::STDAIR_Service& lSTDAIR_Service =
00185         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00186     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00187
00188     // Parse the schedule input file, and generate the Inventories
00189     stdair::BasChronometer lINVGeneration; lINVGeneration.start();
00190     ScheduleParser::generateInventories (iScheduleInputFilename, lBomRoot);
00191     const double lGenerationMeasure = lINVGeneration.elapsed();
00192
00193     // DEBUG
00194     STDAIR_LOG_DEBUG ("Inventory generation time: " << lGenerationMeasure);
00195 }
00196
00197 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00198 void AIRSCHED_Service::
00199 parseAndLoad (const stdair::Filename_T& iScheduleInputFilename,
00200               const stdair::Filename_T& iODInputFilename) {
00201
00202     // First, build the airline inventories from the schedule file
00203     parseAndLoad (iScheduleInputFilename);
00204
00205     // Retrieve the BOM tree root
00206     assert (_airschedServiceContext != NULL);
00207     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00208         *_airschedServiceContext;
00209     stdair::STDAIR_Service& lSTDAIR_Service =
00210         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00211     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00212
00213     // Parse the O&D input file, and generate the O&D periods
00214     stdair::BasChronometer lOnDGeneration; lOnDGeneration.start();
00215     OnDParser::generateOnDPeriods (iODInputFilename, lBomRoot);
00216     const double lGenerationMeasure = lOnDGeneration.elapsed();
00217
00218     // DEBUG
00219     STDAIR_LOG_DEBUG ("O&D generation time: " << lGenerationMeasure);
00220 }
00221
00222 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00223 void AIRSCHED_Service::buildSampleBom() {
00224
00225     // Retrieve the AirSched service context
00226     if (_airschedServiceContext == NULL) {
00227         throw stdair::NonInitialisedServiceException ("The AirSched service has "
00228                                                         "not been initialised");
00229     }
00230     assert (_airschedServiceContext != NULL);

```

```

00231
00232 // Retrieve the AirSched service context and whether it owns the Stdair
00233 // service
00234 AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00235     *_airschedServiceContext;
00236 const bool doesOwnStdairService =
00237     lAIRSCHED_ServiceContext.getOwnStdairServiceFlag();
00238
00239 // Retrieve the StdAir service object from the (AirSched) service context
00240 stdair::STDAIR_Service& lSTDAIR_Service =
00241     lAIRSCHED_ServiceContext.getSTDAIR_Service();
00242
00243 if (doesOwnStdairService == true) {
00244     //
00245     lSTDAIR_Service.buildSampleBom();
00246 }
00247
00248 stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00249 SegmentPathGenerator::createSegmentPathNetwork (lBomRoot);
00250 }
00251
00252 // //////////////////////////////////////
00253 std::string AIRSCHED_Service::
00254 jsonExport (const stdair::AirlineCode_T& iAirlineCode,
00255             const stdair::FlightNumber_T& iFlightNumber,
00256             const stdair::Date_T& iDepartureDate) const {
00257
00258     // Retrieve the AirSched service context
00259     if (_airschedServiceContext == NULL) {
00260         throw stdair::NonInitialisedServiceException ("The AirSched service "
00261             "has not been initialised");
00262     }
00263     assert (_airschedServiceContext != NULL);
00264
00265     // Retrieve the StdAir service object from the (AirSched) service context
00266     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00267         *_airschedServiceContext;
00268     stdair::STDAIR_Service& lSTDAIR_Service =
00269         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00270
00271     // Delegate the JSON export to the dedicated service
00272     return lSTDAIR_Service.jsonExport (iAirlineCode, iFlightNumber,
00273         iDepartureDate);
00274 }
00275
00276 // //////////////////////////////////////
00277 std::string AIRSCHED_Service::csvDisplay() const {
00278
00279     // Retrieve the AirSched service context
00280     if (_airschedServiceContext == NULL) {
00281         throw stdair::NonInitialisedServiceException ("The AirSched service has "
00282             "not been initialised");
00283     }
00284     assert (_airschedServiceContext != NULL);
00285
00286     // Retrieve the STDAIR service object from the (AirSched) service context
00287     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00288         *_airschedServiceContext;
00289     stdair::STDAIR_Service& lSTDAIR_Service =
00290         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00291
00292     // Delegate the BOM building to the dedicated service

```

```

00313     return lSTDAIR_Service.csvDisplay();
00314 }
00315
00316 // //////////////////////////////////////
00317 std::string AIRSCHED_Service::
00318 csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00319             const stdair::FlightNumber_T& iFlightNumber,
00320             const stdair::Date_T& iDepartureDate) const {
00321
00322     // Retrieve the AirSched service context
00323     if (_airschedServiceContext == NULL) {
00324         throw stdair::NonInitialisedServiceException ("The AirSched service has "
00325                                                         "not been initialised");
00326     }
00327     assert (_airschedServiceContext != NULL);
00328
00329     // Retrieve the STDAIR service object from the (AirSched) service context
00330     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00331         *_airschedServiceContext;
00332     stdair::STDAIR_Service& lSTDAIR_Service =
00333         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00334
00335     // Delegate the BOM display to the dedicated service
00336     return lSTDAIR_Service.csvDisplay (iAirlineCode, iFlightNumber,
00337                                         iDepartureDate);
00338 }
00339
00340 // //////////////////////////////////////
00341 void AIRSCHED_Service::simulate() {
00342
00343     // Retrieve the AirSched service context
00344     if (_airschedServiceContext == NULL) {
00345         throw stdair::NonInitialisedServiceException ("The AirSched service has "
00346                                                         "not been initialised");
00347     }
00348     assert (_airschedServiceContext != NULL);
00349
00350     // Retrieve the BOM tree root
00351     AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
00352         *_airschedServiceContext;
00353     stdair::STDAIR_Service& lSTDAIR_Service =
00354         lAIRSCHED_ServiceContext.getSTDAIR_Service();
00355     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00356
00357     // Call the underlying Use Case (command)
00358     stdair::BasChronometer lSimulateChronometer; lSimulateChronometer.start();
00359     Simulator::simulate (lBomRoot);
00360     const double lSimulateMeasure = lSimulateChronometer.elapsed();
00361
00362     // DEBUG
00363     STDAIR_LOG_DEBUG ("Simulation: " << lSimulateMeasure << " - "
00364                       << lAIRSCHED_ServiceContext.display());
00365 }
00366
00367 // //////////////////////////////////////
00368 void AIRSCHED_Service::
00369 buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
00370                       const stdair::BookingRequestStruct& iBookingRequest) {
00371
00372     if (_airschedServiceContext == NULL) {
00373         throw stdair::NonInitialisedServiceException ("The AirSched service has "
00374                                                         "not been initialised");

```

```

00375     }
00376     assert (_airschedServiceContext != NULL);
00377
00378     // Retrieve the BOM tree root
00379     AIRSCHEd_ServiceContext& lAIRSCHEd_ServiceContext =
00380         *_airschedServiceContext;
00381     stdair::STDAIR_Service& lSTDAIR_Service =
00382         lAIRSCHEd_ServiceContext.getSTDAIR_Service();
00383     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00384
00385     // Delegate the call to the dedicated command
00386     stdair::BasChronometer lBuildChronometer; lBuildChronometer.start();
00387     SegmentPathProvider::buildSegmentPathList (ioTravelSolutionList,
00388                                                lBomRoot, iBookingRequest);
00389     const double lBuildMeasure = lBuildChronometer.elapsed();
00390
00391     // DEBUG
00392     STDAIR_LOG_DEBUG ("Segment-path build: " << lBuildMeasure << " - "
00393                     << lAIRSCHEd_ServiceContext.display());
00394 }
00395
00396 }
```

26.140 airsched/service/AIRSCHEd_ServiceContext.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/STDAIR_Service.hpp>
#include <airsched/basic/BasConst_AIRSCHEd_Service.hpp>
#include <airsched/service/AIRSCHEd_ServiceContext.hpp>
```

Namespaces

- namespace [AIRSCHEd](#)

26.141 AIRSCHEd_ServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/STDAIR_Service.hpp>
00009 // AirSched
00010 #include <airsched/basic/BasConst_AIRSCHEd_Service.hpp>
00011 #include <airsched/service/AIRSCHEd_ServiceContext.hpp>
00012
00013 namespace AIRSCHEd {
00014
00015     // //////////////////////////////////////
00016     AIRSCHEd_ServiceContext::AIRSCHEd_ServiceContext ()
```

```

00017     : _ownStdairService (false) {
00018     }
00019
00020     // //////////////////////////////////////
00021     AIRSCHEd_ServiceContext::
00022     AIRSCHEd_ServiceContext (const AIRSCHEd_ServiceContext&) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     AIRSCHEd_ServiceContext::~AIRSCHEd_ServiceContext () {
00028     }
00029
00030     // //////////////////////////////////////
00031     stdair::STDAIR_Service& AIRSCHEd_ServiceContext::getSTDAIR_Service() const {
00032         assert (_stdairService != NULL);
00033         return *_stdairService;
00034     }
00035
00036     // //////////////////////////////////////
00037     const std::string AIRSCHEd_ServiceContext::shortDisplay() const {
00038         std::ostringstream oStr;
00039         oStr << "AIRSCHEd_ServiceContext -- Owns StdAir service: "
00040             << _ownStdairService;
00041         return oStr.str();
00042     }
00043
00044     // //////////////////////////////////////
00045     const std::string AIRSCHEd_ServiceContext::display() const {
00046         std::ostringstream oStr;
00047         oStr << shortDisplay();
00048         return oStr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     const std::string AIRSCHEd_ServiceContext::describe() const {
00053         return shortDisplay();
00054     }
00055
00056     // //////////////////////////////////////
00057     void AIRSCHEd_ServiceContext::reset() {
00058         if (_ownStdairService == true) {
00059             _stdairService.reset();
00060         }
00061     }
00062
00063 }

```

26.142 airsched/service/AIRSCHEd_ServiceContext.hpp File Reference

```

#include <string>

#include <boost/shared_ptr.hpp>

#include <stdair/stdair_service_types.hpp>

#include <stdair/service/ServiceAbstract.hpp>

#include <airsched/AIRSCHEd_Types.hpp>

```

Classes

- class `AIRSCHED::AIRSCHED_ServiceContext`
Class holding the context of the AirSched services.

Namespaces

- namespace `AIRSCHED`

26.143 AIRSCHED_ServiceContext.hpp

```

00001 #ifndef __AIRSCHED_SVC_AIRSCHED_SERVICE_CONTEXT_HPP
00002 #define __AIRSCHED_SVC_AIRSCHED_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 // AirSched
00015 #include <airsched/AIRSCHED_Types.hpp>
00016
00017 namespace AIRSCHED {
00018
00022     class AIRSCHED_ServiceContext : public stdair::ServiceAbstract {
00028         friend class AIRSCHED_Service;
00029         friend class FacAIRSCHEDServiceContext;
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 //////////////////////////////////
00058         void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00059                                 const bool iOwnStdairService) {
00060             _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
00083 private:
00085
00088     AIRSCHED_ServiceContext();
00089
00093     AIRSCHED_ServiceContext (const AIRSCHED_ServiceContext&);
00094
00098     void init();
00099
00103     ~AIRSCHED_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 // __AIRSCHED_SVC_AIRSCHED_SERVICE_CONTEXT_HPP

```

26.144 airsched/service/ServiceAbstract.cpp File Reference

```
#include <airsched/service/ServiceAbstract.hpp>
```

Namespaces

- namespace [AIRSCHED](#)

26.145 ServiceAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // AIRSCHED
00005 #include <airsched/service/ServiceAbstract.hpp>
00006
00007 namespace AIRSCHED {
00008
00009 }

```

26.146 airsched/service/ServiceAbstract.hpp File Reference

```

#include <iostream>
#include <sstream>

```

Classes

- class [AIRSCHED::ServiceAbstract](#)

Namespaces

- namespace [AIRSCHED](#)

Functions

- template<class charT , class traits >
std::basic_ostream< charT, traits > & [operator<<](#) (std::basic_ostream< charT, traits > &ioOut, const [AIRSCHED::ServiceAbstract](#) &iService)
- template<class charT , class traits >
std::basic_istream< charT, traits > & [operator>>](#) (std::basic_istream< charT, traits > &ioln, [AIRSCHED::ServiceAbstract](#) &iService)

26.146.1 Function Documentation

26.146.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream< charT, traits > & *ioOut*, const
[AIRSCHED::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](#).

References [AIRSCHED::ServiceAbstract::toStream\(\)](#).

26.146.1.2 template<class charT , class traits > std::basic_istream<charT,
traits>& operator>> (std::basic_istream< charT, traits > & *ioln*,
[AIRSCHED::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 [AIRSCHED::ServiceAbstract::fromStream\(\)](#).

26.147 ServiceAbstract.hpp

```
00001 #ifndef __AIRSCHED_SERVICEABSTRACT_HPP
00002 #define __AIRSCHED_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
```



```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iostream>
00009 #include <sstream>
00010
00011 namespace AIRSCHED {
00012
00014     class ServiceAbstract {
00015     public:
00016
00018         virtual ~ServiceAbstract() {}
00019
00022         virtual void toStream (std::ostream& ioOut) const {}
00023
00026         virtual void fromStream (std::istream& ioIn) {}
00027
00028     protected:
00030         ServiceAbstract() {}
00031     };
00032 }
00033
00039 template <class charT, class traits>
00040 inline
00041 std::basic_ostream<charT, traits>&
00042 operator<< (std::basic_ostream<charT, traits>& ioOut,
00043           const AIRSCHED::ServiceAbstract& iService) {
00049     std::basic_ostringstream<charT,traits> ostr;
00050     ostr.copyfmt (ioOut);
00051     ostr.width (0);
00052
00053     // Fill string stream
00054     iService.toStream (ostr);
00055
00056     // Print string stream
00057     ioOut << ostr.str();
00058
00059     return ioOut;
00060 }
00061
00067 template <class charT, class traits>
00068 inline
00069 std::basic_istream<charT, traits>&
00070 operator>> (std::basic_istream<charT, traits>& ioIn,
00071           AIRSCHED::ServiceAbstract& ioService) {
00072     // Fill Service object with input stream
00073     ioService.fromStream (ioIn);
00074     return ioIn;
00075 }
00076
00077 #endif // __AIRSCHED_SERVICEABSTRACT_HPP

```

26.148 doc/local/authors.doc File Reference

26.149 doc/local/codingrules.doc File Reference

26.150 doc/local/copyright.doc File Reference

26.151 doc/local/documentation.doc File Reference

26.152 doc/local/features.doc File Reference

26.153 doc/local/help_wanted.doc File Reference

26.154 doc/local/howto_release.doc File Reference

26.155 doc/local/index.doc File Reference

26.156 doc/local/installation.doc File Reference

26.157 doc/local/linking.doc File Reference

26.158 doc/local/test.doc File Reference

26.159 doc/local/users_guide.doc File Reference

26.160 doc/local/verification.doc File Reference

26.161 doc/tutorial/tutorial.doc File Reference

26.162 test/airsched/AirlineScheduleTestSuite.cpp File Reference

26.163 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 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/service/Logger.hpp>
00024 // AirSched
00025 #include <airsched/AIRSCHEM_Service.hpp>
00026 #include <airsched/config/airsched-paths.hpp>
00027
00028 namespace boost_utf = boost::unit_test;
00029
00030 // (Boost) Unit Test XML Report
00031 std::ofstream utfReportStream ("AirlineScheduleTestSuite_utfresults.xml");
00032
```

```

00036 struct UnitTestConfig {
00037     UnitTestConfig() {
00038         boost_utf::unit_test_log.set_stream (utfReportStream);
00039         boost_utf::unit_test_log.set_format (boost_utf::XML);
00040         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00041         //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
00042     ts);
00043     }
00044
00045     ~UnitTestConfig() {
00046     }
00047 };
00048 };
00049
00050
00051 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00052
00053 // Set the UTF configuration (re-direct the output to a specific file)
00054 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00055
00056 // Start the test suite
00057 BOOST_AUTO_TEST_SUITE (master_test_suite)
00058
00059
00060 BOOST_AUTO_TEST_CASE (airsched_simple_inventory_sell) {
00061
00062     // Input file name
00063     const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00064                                                         "/schedule03.csv");
00065
00066     // Output log File
00067     const stdair::Filename_T lLogFilename ("AirlineScheduleTestSuite.log");
00068
00069     // Check that the file path given as input corresponds to an actual file
00070     bool doesExistAndIsReadable =
00071         stdair::BasFileMgr::doesExistAndIsReadable (lScheduleInputFilename);
00072     BOOST_CHECK_MESSAGE (doesExistAndIsReadable == true,
00073                         "The '" << lScheduleInputFilename
00074                         << "' input file can not be open and read");
00075
00076     // Set the log parameters
00077     std::ofstream logOutputFile;
00078     // Open and clean the log outputfile
00079     logOutputFile.open (lLogFilename.c_str());
00080     logOutputFile.clear();
00081
00082     // Instantiate the AirSched service
00083     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00084     AIRSCHED::AIRSCHED_Service airschedService (lLogParams);
00085
00086     // Build the BOM tree from parsing input files
00087     airschedService.parseAndLoad (lScheduleInputFilename);
00088
00089     // Create an empty booking request structure
00090     // \todo: fill the booking request structure from the input parameters
00091     const stdair::AirportCode_T lOrigin ("NCE");
00092     const stdair::AirportCode_T lDestination ("BKK");
00093     const stdair::AirportCode_T lPOS ("NCE");
00094     const stdair::Date_T lPreferredDepartureDate(2007, boost::gregorian::Apr, 21);
00095     const stdair::Date_T lRequestDate (2007, boost::gregorian::Mar, 21);
00096     const stdair::Duration_T lRequestTime (boost::posix_time::hours(8));
00097     const stdair::DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
00098     const stdair::CabinCode_T lPreferredCabin ("Bus");

```

```

00101  const stdair::PartySize_T lPartySize (3);
00102  const stdair::ChannelLabel_T lChannel ("DF");
00103  const stdair::TripType_T lTripType ("RO");
00104  const stdair::DayDuration_T lStayDuration (5);
00105  const stdair::FrequentFlyer_T lFrequentFlyerType ("NONE");
00106  const stdair::Duration_T lPreferredDepartureTime (boost::posix_time::hours(10))
;
00107  const stdair::WTP_T lWTP (2000.0);
00108  const stdair::PriceValue_T lValueOfTime (20.0);
00109  const stdair::BookingRequestStruct lBookingRequest (lOrigin, lDestination,
00110                                                    lPOS,
00111                                                    lPreferredDepartureDate,
00112                                                    lRequestDateTime,
00113                                                    lPreferredCabin,
00114                                                    lPartySize, lChannel,
00115                                                    lTripType, lStayDuration,
00116                                                    lFrequentFlyerType,
00117                                                    lPreferredDepartureTime,
00118                                                    lWTP, lValueOfTime);
00119
00120  //
00121  stdair::TravelSolutionList_T lTravelSolutionList;
00122  airschedService.buildSegmentPathList (lTravelSolutionList, lBookingRequest);
00123  const unsigned int lNbOfTravelSolutions = lTravelSolutionList.size();
00124
00125  // \todo: change the expected number of travel solutions to the actual number
00126  const unsigned int lExpectedNbOfTravelSolutions = 4;
00127
00128  // DEBUG
00129  STDAIR_LOG_DEBUG ("Number of travel solutions for the booking request ' "
00130                  << lBookingRequest.describe() << "' : "
00131                  << lNbOfTravelSolutions << ". It is expected to be "
00132                  << lExpectedNbOfTravelSolutions << ".");
00133
00134  BOOST_CHECK_EQUAL (lNbOfTravelSolutions, lExpectedNbOfTravelSolutions);
00135
00136  BOOST_CHECK_MESSAGE(lNbOfTravelSolutions == lExpectedNbOfTravelSolutions,
00137                    "The number of travel solutions for the booking request ' "
00138                    << lBookingRequest.describe() << "' is equal to "
00139                    << lNbOfTravelSolutions << ", but it should be equal to "
00140                    << lExpectedNbOfTravelSolutions);
00141
00142  // Close the Log outputFile
00143  logOutputFile.close();
00144 }
00145
00146 // End the test suite
00147 BOOST_AUTO_TEST_SUITE_END()
00148
00149

```

26.164 test/airsched/AirlineScheduleTestSuite.hpp File Reference

```

#include <sstream>

#include <cppunit/extensions/HelperMacros.h>

```

Classes

- class [AirlineScheduleTestSuite](#)

Functions

- [CPPUNIT_TEST_SUITE_REGISTRATION](#) ([AirlineScheduleTestSuite](#))

26.164.1 Function Documentation

26.164.1.1 CPPUNIT_TEST_SUITE_REGISTRATION(AirlineScheduleTestSuite)

26.165 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);
```

Index

~AIRSCHED_Service
AIRSCHED::AIRSCHED_Service, 152
~FacAIRSCHEDServiceContext
AIRSCHED::FacAIRSCHEDServiceContext, AIRSCHED::OnDParserHelper::doEndOnD, 173
~FacServiceAbstract
AIRSCHED::FacServiceAbstract, 175
~FlagSaver
AIRSCHED::FlagSaver, 178
~OriginDestinationSet
AIRSCHED::OriginDestinationSet, 204
~OriginDestinationSetKey
AIRSCHED::OriginDestinationSetKey, 208
~ReachableUniverse
AIRSCHED::ReachableUniverse, 218
~ReachableUniverseKey
AIRSCHED::ReachableUniverseKey, 222
~SegmentPathPeriod
AIRSCHED::SegmentPathPeriod, 233
~SegmentPathPeriodKey
AIRSCHED::SegmentPathPeriodKey, 240
~ServiceAbstract
AIRSCHED::ServiceAbstract, 247
_airlineCode
AIRSCHED::FlightPeriodStruct, 185
AIRSCHED::OnDPeriodStruct, 201
_airlineCodeList
AIRSCHED::OnDPeriodStruct, 201
_airlineList
airsched::SearchString_T, 226
_airportList
AIRSCHED::FlightPeriodStruct, 187
_airportOrderedList
AIRSCHED::FlightPeriodStruct, 187
_areSegmentDefinitionsSpecific
AIRSCHED::FlightPeriodStruct, 187
_boardingDate
AIRSCHED::SegmentStruct, 246
_boardingDateOffset
AIRSCHED::LegStruct, 192
_boardingPoint
AIRSCHED::LegStruct, 192
AIRSCHED::SegmentStruct, 246
_boardingTime
AIRSCHED::LegStruct, 192
AIRSCHED::SegmentStruct, 246
_bomRoot
AIRSCHED::OnDParserHelper::doEndOnD, 172
AIRSCHED::OnDParserHelper::OnDParser, 196
AIRSCHED::ScheduleParserHelper::doEndFlight, 170
AIRSCHED::ScheduleParserHelper::FlightPeriodParser, 181
_cabinCode
AIRSCHED::LegCabinStruct, 190
AIRSCHED::SegmentCabinStruct, 229
_cabinList
AIRSCHED::LegStruct, 193
AIRSCHED::SegmentStruct, 246
_capacity
AIRSCHED::LegCabinStruct, 190
_classCode
AIRSCHED::OnDPeriodStruct, 201
_classCodeList
AIRSCHED::OnDPeriodStruct, 201
_classes
AIRSCHED::FareFamilyStruct, 178
AIRSCHED::SegmentCabinStruct, 229
_code
airsched::Airline_T, 149
airsched::Place_T, 216
_date
airsched::Date_T, 159
_dateList
airsched::SearchString_T, 226
_dateOffset
AIRSCHED::FlightPeriodStruct, 186
_datePeriod
AIRSCHED::OnDPeriodStruct, 200
_dateRange
AIRSCHED::FlightPeriodStruct, 185
_dateRangeEnd
AIRSCHED::FlightPeriodStruct, 186
AIRSCHED::OnDPeriodStruct, 201
_dateRangeStart
AIRSCHED::FlightPeriodStruct, 186
AIRSCHED::OnDPeriodStruct, 201
_day

- airsched::Date_T, [159](#)
- _describeKey
 - AirlineScheduleTestSuite, [150](#)
- _destination
 - AIRSCHED::OnDPeriodStruct, [200](#)
- _dow
 - AIRSCHED::FlightPeriodStruct, [185](#)
- _elapsed
 - AIRSCHED::LegStruct, [193](#)
 - AIRSCHED::SegmentStruct, [246](#)
- _familyCode
 - AIRSCHED::FareFamilyStruct, [177](#)
- _fareFamilies
 - AIRSCHED::SegmentCabinStruct, [229](#)
- _flightNumber
 - AIRSCHED::FlightPeriodStruct, [185](#)
- _flightPeriod
 - AIRSCHED::ScheduleParserHelper::doEndFlight, [170](#)
 - AIRSCHED::ScheduleParserHelper::FlightPeriod, [181](#)
 - AIRSCHED::ScheduleParserHelper::ParsePrimary, [213](#)
 - AIRSCHED::ScheduleParserHelper::storeAirlineCode, [261](#)
 - AIRSCHED::ScheduleParserHelper::storeBoardingTime, [262](#)
 - AIRSCHED::ScheduleParserHelper::storeCapacity, [264](#)
 - AIRSCHED::ScheduleParserHelper::storeClass, [266](#)
 - AIRSCHED::ScheduleParserHelper::storeDateRangeEnd, [268](#)
 - AIRSCHED::ScheduleParserHelper::storeDateRangeStart, [271](#)
 - AIRSCHED::ScheduleParserHelper::storeDays, [275](#)
 - AIRSCHED::ScheduleParserHelper::storeElapsedTime, [277](#)
 - AIRSCHED::ScheduleParserHelper::storeSegmentCode, [280](#)
 - AIRSCHED::ScheduleParserHelper::storeSegmentCabin, [281](#)
 - AIRSCHED::ScheduleParserHelper::storeFlightNumber, [283](#)
 - AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint, [284](#)
 - AIRSCHED::ScheduleParserHelper::storeLegClass, [286](#)
- AIRSCHED::Airline_T, [149](#)
- AIRSCHED::FlightPeriodStruct, [186](#)
- AIRSCHED::OnDPeriodStruct, [202](#)
- AIRSCHED::SegmentCabinStruct, [229](#)
- AIRSCHED::FlightPeriodStruct, [187](#)
- AIRSCHED::OnDPeriodStruct, [202](#)
- AIRSCHED::FlightPeriodStruct, [186](#)
- AIRSCHED::FlightPeriodStruct, [186](#)
- AIRSCHED::FlightPeriodStruct, [187](#)
- AIRSCHED::OnDPeriodStruct, [202](#)
- AIRSCHED::FlightPeriodStruct, [186](#)
- AIRSCHED::OnDPeriodStruct, [202](#)
- AIRSCHED::FlightPeriodStruct, [187](#)
- AIRSCHED::FlightPeriodStruct, [188](#)
- AIRSCHED::FlightPeriodStruct, [186](#)
- AIRSCHED::OnDPeriodStruct, [202](#)
- AIRSCHED::OriginDestinationSet, [206](#)
- AIRSCHED::ReachableUniverse, [220](#)
- AIRSCHED::SegmentPathPeriod, [238](#)
- AIRSCHED::ScheduleParserHelper::storeLegOffPoint, [287](#)
- AIRSCHED::ScheduleParserHelper::storeOffTime, [289](#)
- AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint, [292](#)
- AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode, [293](#)
- AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint, [295](#)
- AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity, [296](#)
- _holderMap
 - AIRSCHED::OriginDestinationSet, [206](#)
 - AIRSCHED::ReachableUniverse, [220](#)
 - AIRSCHED::SegmentPathPeriod, [238](#)
- _isPreferred

- [_labels](#)
 - [airsched::Passenger_T, 214](#)
- [_legAlreadyDefined](#)
 - [AIRSCHED::FlightPeriodStruct, 185](#)
- [_legList](#)
 - [AIRSCHED::FlightPeriodStruct, 185](#)
- [_month](#)
 - [airsched::Date_T, 160](#)
- [_name](#)
 - [airsched::Airline_T, 149](#)
 - [airsched::Place_T, 216](#)
- [_nbOfAirlines](#)
 - [AIRSCHED::OnDPeriodStruct, 201](#)
- [_number](#)
 - [airsched::Passenger_T, 215](#)
- [_offDate](#)
 - [AIRSCHED::SegmentStruct, 246](#)
- [_offDateOffset](#)
 - [AIRSCHED::LegStruct, 193](#)
- [_offPoint](#)
 - [AIRSCHED::LegStruct, 192](#)
 - [AIRSCHED::SegmentStruct, 246](#)
- [_offTime](#)
 - [AIRSCHED::LegStruct, 193](#)
 - [AIRSCHED::SegmentStruct, 246](#)
- [_onDPeriod](#)
 - [AIRSCHED::OnDParserHelper::doEndOfSegmentList, 172](#)
 - [AIRSCHED::OnDParserHelper::OnDParserSegmentPathPeriodListList, 196](#)
 - [AIRSCHED::OnDParserHelper::ParserSegmentRangeEnd, 211](#)
 - [AIRSCHED::OnDParserHelper::storeAirlineCode, 259](#)
 - [AIRSCHED::OnDParserHelper::storeClassCode, 265](#)
 - [AIRSCHED::OnDParserHelper::storeDateRangeEnd, 270](#)
 - [AIRSCHED::OnDParserHelper::storeDateRangeStager, 272](#)
 - [AIRSCHED::OnDParserHelper::storeDestinationPlace, 274](#)
 - [AIRSCHED::OnDParserHelper::storeEndTime, 278](#)
 - [AIRSCHED::OnDParserHelper::storeOrigin, 290](#)
 - [AIRSCHED::OnDParserHelper::storeStartRangeTime, 298](#)
- [_origin](#)
 - [AIRSCHED::OnDPeriodStruct, 200](#)
- [_parent](#)
 - [AIRSCHED::OriginDestinationSet, 206](#)
 - [AIRSCHED::ReachableUniverse, 220](#)
 - [AIRSCHED::SegmentPathPeriod, 238](#)
- [_passengerList](#)
 - [airsched::SearchString_T, 226](#)
- [_placeList](#)
 - [airsched::SearchString_T, 226](#)
- [_pool](#)
 - [AIRSCHED::FacServiceAbstract, 176](#)
- [_reldays](#)
 - [airsched::Date_T, 159](#)
- [_searchString](#)
 - [airsched::SearchStringParser, 228](#)
 - [airsched::store_adult_passenger_type, 250](#)
 - [airsched::store_airline_code, 251](#)
 - [airsched::store_airline_name, 252](#)
 - [airsched::store_airline_sign, 253](#)
 - [airsched::store_child_passenger_type, 254](#)
 - [airsched::store_date, 255](#)
 - [airsched::store_passenger_number, 256](#)
 - [airsched::store_pet_passenger_type, 257](#)
 - [airsched::store_place_element, 258](#)
- [_segmentList](#)
 - [AIRSCHED::FlightPeriodStruct, 185](#)
- [_segmentPathPeriodListList](#)
 - [AIRSCHED::ReachableUniverse, 221](#)
- [_timeRangeEnd](#)
 - [AIRSCHED::OnDPeriodStruct, 201](#)
- [_timeRangeStart](#)
 - [AIRSCHED::OnDPeriodStruct, 201](#)
- [_type](#)
 - [airsched::SearchString_T, 227](#)
 - [airsched::SearchString_T, 227](#)
 - [airsched::SearchString_T, 227](#)
 - [airsched::SearchString_T, 227](#)
 - [airsched::SearchString_T, 226](#)
- [_type](#)
 - [airsched::Passenger_T, 215](#)
- [_year](#)
 - [airsched::Date_T, 160](#)
- [addAirport](#)
 - [AIRSCHED::FlightPeriodStruct, 183](#)
- [addFareFamily](#)

- AIRSCHED::FlightPeriodStruct, 184, 185
- addSegmentCabin
 - AIRSCHED::FlightPeriodStruct, 184
- ADULT
 - airsched::Passenger_T, 214
- airline_code
 - AIRSCHED::ScheduleParserHelper::FlightPeriodDefinition, 162
 - airsched::SearchStringParser::definition, 166
- airline_code_p
 - AIRSCHED::OnDParserHelper, 142
 - AIRSCHED::ScheduleParserHelper, 145
- airline_name
 - airsched::SearchStringParser::definition, 166
- Airline_T
 - airsched::Airline_T, 149
- AirlineList_T
 - airsched, 133
- AirlineScheduleTestSuite, 149
 - _describeKey, 150
 - AirlineScheduleTestSuite, 150
 - externalMemoryManagement, 150
 - scheduleParsing, 150
- AirlineScheduleTestSuite.hpp
 - CPPUNIT_TEST_SUITE_REGISTRATION, 462
- airport_p
 - AIRSCHED::OnDParserHelper, 142
 - AIRSCHED::ScheduleParserHelper, 146
- AirportList_T
 - AIRSCHED, 139
- AirportOrderedList_T
 - AIRSCHED, 139
- AIRSCHED, 135
 - AirportList_T, 139
 - AirportOrderedList_T, 139
 - AIRSCHED_ServicePtr_T, 137
 - bounded1_4_p_t, 139
 - bounded2_p_t, 139
 - bounded4_p_t, 139
 - char_t, 137
 - chset_t, 138
 - DateOffsetList_T, 140
 - DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION, 141
 - FareFamilyStructList_T, 139
 - int1_p_t, 138
 - iterator_t, 138
 - LegCabinStructList_T, 139
 - LegStructList_T, 139
 - OriginDestinationSet::serialize< ba::text_iarchive >, 141
 - OriginDestinationSet::serialize< ba::text_oarchive >, 141
 - OriginDestinationSetKey::serialize< ba::text_iarchive >, 141
 - OriginDestinationSetKey::serialize< ba::text_oarchive >, 141
 - OriginDestinationSetList_T, 139
 - OriginDestinationSetMap_T, 139
 - ReachableUniverse::serialize< ba::text_iarchive >, 141
 - ReachableUniverse::serialize< ba::text_oarchive >, 141
 - ReachableUniverseKey::serialize< ba::text_iarchive >, 141
 - ReachableUniverseKey::serialize< ba::text_oarchive >, 141
 - ReachableUniverseList_T, 140
 - ReachableUniverseMap_T, 140
 - repeat_p_t, 138
 - rule_t, 138
 - scanner_t, 138
 - SegmentCabinStructList_T, 140
 - SegmentPathPeriodLightList_T, 140
 - SegmentPathPeriodList_T, 140
 - SegmentPathPeriodListList_T, 140
 - SegmentPathPeriodMultimap_T, 140
 - SegmentStructList_T, 141
 - uint1_4_p_t, 138
 - uint2_p_t, 138
 - uint4_p_t, 138
- airsched, 132
 - AirlineList_T, 133
 - DateList_T, 133
 - int1_p, 134
 - parseBookingRequest, 133
 - PassengerList_T, 133
 - PlaceList_T, 133
 - uint1_2_p, 134
 - uint1_4_p, 134
 - uint1_p, 134
 - uint2_4_p, 134
 - uint2_p, 134
 - uint4_p, 134
- airsched.cpp
 - createStringFromWordList, 308

K_AIRSCHED_DEFAULT_BOOKING_-airsched/bom/OriginDestinationSetKey.cpp,
 REQUEST, 308 350, 351
 K_AIRSCHED_DEFAULT_BOOKING_-airsched/bom/OriginDestinationSetKey.hpp,
 REQUEST_MODE, 309 352
 K_AIRSCHED_DEFAULT_BUILT_IN_-airsched/bom/OriginDestinationSetTypes.hpp,
 INPUT, 309 354
 K_AIRSCHED_DEFAULT_INPUT_FILENAME_-airsched/bom/ReachableUniverse.cpp, 355
 308 airsched/bom/ReachableUniverse.hpp, 356,
 K_AIRSCHED_DEFAULT_LOG_FILENAME, 357
 308 airsched/bom/ReachableUniverseKey.cpp,
 K_AIRSCHED_EARLY_RETURN_STATUS, 359
 309 airsched/bom/ReachableUniverseKey.hpp,
 main, 309 361
 operator<=, 308 airsched/bom/ReachableUniverseTypes.hpp,
 parseBookingRequest, 309 363
 readConfiguration, 308 airsched/bom/SegmentCabinStruct.cpp, 363,
 WordList_T, 308 364
 airsched/ Directory Reference, 129 airsched/bom/SegmentCabinStruct.hpp, 364,
 airsched/AIRSCHED_Service.hpp, 300 365
 airsched/AIRSCHED_Types.hpp, 302 airsched/bom/SegmentPathPeriod.cpp, 366
 airsched/basic/ Directory Reference, 129 airsched/bom/SegmentPathPeriod.hpp, 371,
 airsched/basic/BasConst.cpp, 303 372
 airsched/basic/BasConst_AIRSCHED_Service.cpp, 374, 375
 304 airsched/bom/SegmentPathPeriodKey.cpp,
 airsched/basic/BasConst_General.hpp, 304 airsched/bom/SegmentPathPeriodKey.hpp,
 airsched/basic/BasParserTypes.hpp, 305, 306 376, 377
 airsched/batches/ Directory Reference, 130 airsched/bom/SegmentPathPeriodTypes.hpp,
 airsched/batches/airsched.cpp, 307, 310 379, 380
 airsched/batches/BookingRequestParser.cpp, 381
 316, 318
 airsched/batches/BookingRequestParser.hpp, 382
 324 airsched/bom/SegmentStruct.cpp, 383
 airsched/bom/ Directory Reference, 130 airsched/bom/SegmentStruct.hpp, 384, 385
 airsched/bom/AirportList.hpp, 327 airsched/command/ Directory Reference, 131
 airsched/bom/BomDisplay.cpp, 327, 328 airsched/command/InventoryGenerator.cpp,
 airsched/bom/BomDisplay.hpp, 329, 330 385, 386
 airsched/bom/FareFamilyStruct.cpp, 330 airsched/command/InventoryGenerator.hpp,
 airsched/bom/FareFamilyStruct.hpp, 331 388
 airsched/bom/FlightPeriodStruct.cpp, 332 airsched/command/OnDParser.cpp, 389
 airsched/bom/FlightPeriodStruct.hpp, 336, 390
 337 airsched/command/OnDParserHelper.cpp,
 airsched/bom/LegCabinStruct.cpp, 338, 339 391, 392
 airsched/bom/LegCabinStruct.hpp, 339, 340 airsched/command/OnDParserHelper.hpp,
 airsched/bom/LegStruct.cpp, 340 398, 399
 airsched/bom/LegStruct.hpp, 342 airsched/command/OnDPeriodGenerator.cpp,
 airsched/bom/OnDPeriodStruct.cpp, 343 401, 402
 airsched/bom/OnDPeriodStruct.hpp, 345 airsched/command/OnDPeriodGenerator.hpp,
 airsched/bom/OriginDestinationSet.cpp, 346 402, 403
 airsched/bom/OriginDestinationSet.hpp, 348 airsched/command/ScheduleParser.cpp, 403,
 404

- airsched/command/ScheduleParser.hpp, [404](#), [405](#)
- airsched/command/ScheduleParserHelper.cpp, [405](#), [406](#)
- airsched/command/ScheduleParserHelper.hpp, [418](#), [419](#)
- airsched/command/SegmentPathGenerator.cpp, [422](#)
- airsched/command/SegmentPathGenerator.hpp, [429](#), [430](#)
- airsched/command/SegmentPathProvider.cpp, [431](#)
- airsched/command/SegmentPathProvider.hpp, [434](#)
- airsched/command/Simulator.cpp, [435](#)
- airsched/command/Simulator.hpp, [436](#)
- airsched/command/TravelSolutionParser.cpp, [437](#)
- airsched/command/TravelSolutionParser.hpp, [440](#)
- airsched/config/ Directory Reference, [131](#)
- airsched/config/airsched-paths.hpp, [442](#)
- airsched/config/airsched-paths.hpp.in, [442](#)
- airsched/factory/ Directory Reference, [131](#)
- airsched/factory/FacAIRSCHEDServiceContext.cpp, [443](#)
- airsched/factory/FacAIRSCHEDServiceContext.hpp, [444](#)
- airsched/factory/FacServiceAbstract.cpp, [445](#)
- airsched/factory/FacServiceAbstract.hpp, [445](#), [446](#)
- airsched/service/ Directory Reference, [132](#)
- airsched/service/AIRSCHED_Service.cpp, [446](#), [447](#)
- airsched/service/AIRSCHED_ServiceContext.cpp, [453](#)
- airsched/service/AIRSCHED_ServiceContext.hpp, [454](#), [455](#)
- airsched/service/ServiceAbstract.cpp, [456](#)
- airsched/service/ServiceAbstract.hpp, [456](#), [457](#)
- airsched::Airline_T, [148](#)
 - _code, [149](#)
 - _isPreferred, [149](#)
 - _name, [149](#)
 - Airline_T, [149](#)
 - display, [149](#)
- AIRSCHED::AIRSCHED_Service, [151](#)
 - ~AIRSCHED_Service, [152](#)
 - AIRSCHED_Service, [151](#), [152](#)
- buildSampleBom, [153](#)
- buildSegmentPathList, [153](#)
- csvDisplay, [154](#)
- jsonExport, [154](#)
- parseAndLoad, [152](#), [153](#)
- simulate, [154](#)
- AIRSCHED::AIRSCHED_ServiceContext, [155](#)
- AIRSCHED_Service, [155](#)
- FacAIRSCHEDServiceContext, [155](#)
- AIRSCHED::BomDisplay, [156](#)
- csvDisplay, [156](#), [157](#)
- airsched::Date_T, [158](#)
 - _date, [159](#)
 - _day, [159](#)
 - _month, [160](#)
 - _reldays, [159](#)
 - _year, [160](#)
- Date_T, [159](#)
- display, [159](#)
- getDate, [159](#)
- AIRSCHED::FacAIRSCHEDServiceContext, [173](#)
- ~FacAIRSCHEDServiceContext, [173](#)
- create, [174](#)
- FacAIRSCHEDServiceContext, [173](#)
- instance, [174](#)
- AIRSCHED::FacServiceAbstract, [174](#)
- ~FacServiceAbstract, [175](#)
- _pool, [176](#)
- clean, [176](#)
- FacServiceAbstract, [175](#)
- ServicePool_T, [175](#)
- AIRSCHED::FareFamilyStruct, [176](#)
 - _classes, [178](#)
- _familyCode, [177](#)
- describe, [177](#)
- FareFamilyStruct, [177](#)
- AIRSCHED::FlagSaver, [178](#)
 - ~FlagSaver, [178](#)
 - FlagSaver, [178](#)
- AIRSCHED::FlightPeriodFileParser, [179](#)
- FlightPeriodFileParser, [179](#)
- generateInventories, [180](#)
- AIRSCHED::FlightPeriodStruct, [181](#)
 - _airlineCode, [185](#)
 - _airportList, [187](#)
 - _airportOrderedList, [187](#)
 - _areSegmentDefinitionsSpecific, [187](#)
 - _dateOffset, [186](#)
 - _dateRange, [185](#)

- [_dateRangeEnd, 186](#)
- [_dateRangeStart, 186](#)
- [_dow, 185](#)
- [_flightNumber, 185](#)
- [_itDay, 186](#)
- [_itHours, 187](#)
- [_itLeg, 186](#)
- [_itLegCabin, 186](#)
- [_itMinutes, 187](#)
- [_itMonth, 186](#)
- [_itSeconds, 187](#)
- [_itSegment, 187](#)
- [_itSegmentCabin, 188](#)
- [_itYear, 186](#)
- [_legAlreadyDefined, 185](#)
- [_legList, 185](#)
- [_segmentList, 185](#)
- [addAirport, 183](#)
- [addFareFamily, 184, 185](#)
- [addSegmentCabin, 184](#)
- [buildSegments, 184](#)
- [describe, 183](#)
- [FlightPeriodStruct, 183](#)
- [getDate, 183](#)
- [getTime, 183](#)
- [AIRSCHED::InventoryGenerator, 188](#)
- [FFFlightPeriodFileParser, 189](#)
- [FlightPeriodFileParser, 189](#)
- [ScheduleParser, 189](#)
- [ScheduleParserHelper::doEndFlight, 189](#)
- [AIRSCHED::LegCabinStruct, 189](#)
- [_cabinCode, 190](#)
- [_capacity, 190](#)
- [describe, 190](#)
- [fill, 190](#)
- [AIRSCHED::LegStruct, 191](#)
- [_boardingDateOffset, 192](#)
- [_boardingPoint, 192](#)
- [_boardingTime, 192](#)
- [_cabinList, 193](#)
- [_elapsed, 193](#)
- [_offDateOffset, 193](#)
- [_offPoint, 192](#)
- [_offTime, 193](#)
- [describe, 192](#)
- [fill, 192](#)
- [LegStruct, 192](#)
- [AIRSCHED::OnDInputFileNotFoundExceptioA](#)
- [193](#)
- [OnDInputFileNotFoundExceptio](#)
- [194](#)
- [AIRSCHED::OnDParser, 194](#)
- [generateOnDPeriods, 194](#)
- [AIRSCHED::OnDParserHelper, 141](#)
- [airline_code_p, 142](#)
- [airport_p, 142](#)
- [alpha_cap_set_p, 142](#)
- [class_code_p, 143](#)
- [day_p, 143](#)
- [hours_p, 143](#)
- [minutes_p, 143](#)
- [month_p, 143](#)
- [seconds_p, 143](#)
- [uint1_4_p, 144](#)
- [uint2_p, 144](#)
- [uint4_p, 144](#)
- [year_p, 143](#)
- [AIRSCHED::OnDParserHelper::doEndOnD,](#)
- [171](#)
- [_bomRoot, 172](#)
- [_onDPeriod, 172](#)
- [doEndOnD, 172](#)
- [operator\(\), 172](#)
- [AIRSCHED::OnDParserHelper::OnDParser,](#)
- [195](#)
- [_bomRoot, 196](#)
- [_onDPeriod, 196](#)
- [OnDParser, 196](#)
- [AIRSCHED::OnDParserHelper::OnDParser::definition,](#)
- [167](#)
- [date, 169](#)
- [definition, 168](#)
- [ond, 168](#)
- [ond_end, 169](#)
- [ond_key, 169](#)
- [ond_list, 168](#)
- [segment, 169](#)
- [start, 168](#)
- [time, 169](#)
- [AIRSCHED::OnDParserHelper::ParserSemanticAction,](#)
- [210](#)
- [_onDPeriod, 211](#)
- [ParserSemanticAction, 210](#)
- [AIRSCHED::OnDParserHelper::storeAirlineCode,](#)
- [258](#)
- [_onDPeriod, 259](#)
- [operator\(\), 259](#)
- [storeAirlineCode, 259](#)
- [AIRSCHED::OnDParserHelper::storeClassCode,](#)
- [264](#)
- [_onDPeriod, 265](#)

- operator(), 265
- storeClassCode, 265
- AIRSCHEd::OnDParserHelper::storeDateRangeEnd, 268
- _onDPeriod, 270
- operator(), 269
- storeDateRangeEnd, 269
- AIRSCHEd::OnDParserHelper::storeDateRangeStart, 271
- _onDPeriod, 272
- operator(), 272
- storeDateRangeStart, 272
- AIRSCHEd::OnDParserHelper::storeDestination, 273
- _onDPeriod, 274
- operator(), 274
- storeDestination, 273
- AIRSCHEd::OnDParserHelper::storeEndRangeTime, 277
- _onDPeriod, 278
- operator(), 278
- storeEndRangeTime, 278
- AIRSCHEd::OnDParserHelper::storeOrigin, 289
- _onDPeriod, 290
- operator(), 290
- storeOrigin, 290
- AIRSCHEd::OnDParserHelper::storeStartRangeTime, 297
- _onDPeriod, 298
- operator(), 298
- storeStartRangeTime, 297
- AIRSCHEd::OnDPeriodFileParser, 196
- generateOnDPeriods, 197
- OnDPeriodFileParser, 197
- AIRSCHEd::OnDPeriodGenerator, 197
- OnDParser, 198
- OnDParserHelper::doEndOnD, 198
- OnDPeriodFileParser, 198
- AIRSCHEd::OnDPeriodStruct, 198
- _airlineCode, 201
- _airlineCodeList, 201
- _classCode, 201
- _classCodeList, 201
- _datePeriod, 200
- _dateRangeEnd, 201
- _dateRangeStart, 201
- _destination, 200
- _itDay, 202
- _itHours, 202
- _itMinutes, 202
- _itMonth, 202
- _itSeconds, 202
- _itYear, 202
- _nbOfAirlines, 201
- _origin, 200
- _timeRangeEnd, 201
- _timeRangeStart, 201
- describe, 200
- describeTSKey, 200
- getDate, 199
- getFirstAirlineCode, 199
- getTime, 200
- OnDPeriodStruct, 199
- AIRSCHEd::OriginDestinationSet, 202
- ~OriginDestinationSet, 204
- _holderMap, 206
- key, 206
- _parent, 206
- boost::serialization::access, 206
- describeKey, 205
- fromStream, 205
- getDestination, 204
- getHolderMap, 205
- getKey, 204
- getParent, 205
- Key_T, 204
- OriginDestinationSet, 204
- serialize, 206
- stdair::FacBom, 206
- stdair::FacBomManager, 206
- toStream, 205
- toString, 205
- AIRSCHEd::OriginDestinationSetKey, 207
- ~OriginDestinationSetKey, 208
- boost::serialization::access, 209
- fromStream, 208
- getOffPoint, 208
- OriginDestinationSetKey, 208
- serialize, 209
- toStream, 208
- toString, 209
- airsched::Passenger_T
- ADULT, 214
- CHILD, 214
- LAST_VALUE, 214
- PET, 214
- airsched::Passenger_T, 213
- _labels, 214
- _number, 215

- [_type](#), [215](#)
 - [display](#), [214](#)
 - [Passenger_T](#), [214](#)
 - [PassengerType_T](#), [214](#)
- [airsched::Place_T](#), [215](#)
 - [_code](#), [216](#)
 - [_name](#), [216](#)
 - [display](#), [216](#)
 - [Place_T](#), [215](#)
- [AIRSCHED::ReachableUniverse](#), [216](#)
 - [~ReachableUniverse](#), [218](#)
 - [_holderMap](#), [220](#)
 - [_key](#), [220](#)
 - [_parent](#), [220](#)
 - [_segmentPathPeriodListList](#), [221](#)
 - [boost::serialization::access](#), [220](#)
 - [describeKey](#), [219](#)
 - [fromStream](#), [219](#)
 - [getHolderMap](#), [218](#)
 - [getKey](#), [218](#)
 - [getOrigin](#), [218](#)
 - [getParent](#), [218](#)
 - [getSegmentPathPeriodListList](#), [218](#)
 - [Key_T](#), [217](#)
 - [ReachableUniverse](#), [218](#)
 - [SegmentPathGenerator](#), [220](#)
 - [serialize](#), [219](#)
 - [stdair::FacBom](#), [220](#)
 - [stdair::FacBomManager](#), [220](#)
 - [toStream](#), [219](#)
 - [toString](#), [219](#)
- [AIRSCHED::ReachableUniverseKey](#), [221](#)
 - [~ReachableUniverseKey](#), [222](#)
 - [boost::serialization::access](#), [223](#)
 - [fromStream](#), [223](#)
 - [getBoardingPoint](#), [222](#)
 - [ReachableUniverseKey](#), [222](#)
 - [serialize](#), [223](#)
 - [toStream](#), [222](#)
 - [toString](#), [223](#)
- [AIRSCHED::ScheduleInputFileNotFoundException](#), [223](#)
 - [ScheduleInputFileNotFoundException](#), [224](#)
- [AIRSCHED::ScheduleParser](#), [224](#)
 - [generateInventories](#), [225](#)
- [AIRSCHED::ScheduleParserHelper](#), [144](#)
 - [airline_code_p](#), [145](#)
 - [airport_p](#), [146](#)
 - [cabin_code_p](#), [146](#)
 - [class_code_list_p](#), [147](#)
 - [day_p](#), [146](#)
 - [dow_p](#), [146](#)
 - [family_code_p](#), [147](#)
 - [flight_number_p](#), [145](#)
 - [hours_p](#), [146](#)
 - [int1_p](#), [147](#)
 - [minutes_p](#), [146](#)
 - [month_p](#), [145](#)
 - [seconds_p](#), [146](#)
 - [uint1_4_p](#), [147](#)
 - [uint2_p](#), [147](#)
 - [uint4_p](#), [147](#)
 - [year_p](#), [145](#)
- [AIRSCHED::ScheduleParserHelper::doEndFlight](#), [169](#)
 - [_bomRoot](#), [170](#)
 - [_flightPeriod](#), [170](#)
 - [doEndFlight](#), [170](#)
 - [operator\(\)](#), [170](#)
- [AIRSCHED::ScheduleParserHelper::FlightPeriodParser](#), [180](#)
 - [_bomRoot](#), [181](#)
 - [_flightPeriod](#), [181](#)
 - [FlightPeriodParser](#), [181](#)
- [AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition](#), [160](#)
 - [airline_code](#), [162](#)
 - [date](#), [162](#)
 - [date_offset](#), [162](#)
 - [definition](#), [161](#)
 - [dow](#), [162](#)
 - [family_cabin_details](#), [164](#)
 - [flight_key](#), [162](#)
 - [flight_number](#), [162](#)
 - [flight_period](#), [161](#)
 - [flight_period_end](#), [162](#)
 - [flight_period_list](#), [161](#)
 - [full_family_cabin_details](#), [164](#)
 - [full_segment_cabin_details](#), [163](#)
 - [generic_segment](#), [164](#)
 - [leg](#), [163](#)
 - [leg_cabin_details](#), [163](#)
 - [leg_details](#), [163](#)
 - [leg_key](#), [163](#)
 - [segment_cabin_details](#), [163](#)
 - [segment_key](#), [163](#)
 - [segment_section](#), [163](#)
 - [specific_segment_list](#), [164](#)
 - [start](#), [161](#)

- time, 162
- AIRSCHED::ScheduleParserHelper::ParserSemanticAction, 211
- _flightPeriod, 213
- ParserSemanticAction, 212
- AIRSCHED::ScheduleParserHelper::storeAirlineCode, 260
- _flightPeriod, 261
- operator(), 260
- storeAirlineCode, 260
- AIRSCHED::ScheduleParserHelper::storeBoardingTime, 261
- _flightPeriod, 262
- operator(), 262
- storeBoardingTime, 262
- AIRSCHED::ScheduleParserHelper::storeCapacity, 263
- _flightPeriod, 264
- operator(), 263
- storeCapacity, 263
- AIRSCHED::ScheduleParserHelper::storeClasses, 265
- _flightPeriod, 266
- operator(), 266
- storeClasses, 266
- AIRSCHED::ScheduleParserHelper::storeDateRangeEnd, 267
- _flightPeriod, 268
- operator(), 268
- storeDateRangeEnd, 268
- AIRSCHED::ScheduleParserHelper::storeDateRangeStart, 270
- _flightPeriod, 271
- operator(), 271
- storeDateRangeStart, 271
- AIRSCHED::ScheduleParserHelper::storeDow, 274
- _flightPeriod, 275
- operator(), 275
- storeDow, 275
- AIRSCHED::ScheduleParserHelper::storeElapsedTime, 276
- _flightPeriod, 277
- operator(), 277
- storeElapsedTime, 276
- AIRSCHED::ScheduleParserHelper::storeFamilyCode, 279
- _flightPeriod, 280
- operator(), 279
- storeFamilyCode, 279
- AIRSCHED::ScheduleParserHelper::storeFClasses, 280
- _flightPeriod, 281
- operator(), 281
- storeFClasses, 281
- AIRSCHED::ScheduleParserHelper::storeFlightNumber, 282
- _flightPeriod, 283
- operator(), 283
- storeFlightNumber, 282
- AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint, 283
- _flightPeriod, 284
- operator(), 284
- storeLegBoardingPoint, 284
- AIRSCHED::ScheduleParserHelper::storeLegCabinCode, 285
- _flightPeriod, 286
- operator(), 286
- storeLegCabinCode, 285
- AIRSCHED::ScheduleParserHelper::storeLegOffPoint, 286
- _flightPeriod, 287
- operator(), 287
- storeLegOffPoint, 287
- AIRSCHED::ScheduleParserHelper::storeOffTime, 288
- _flightPeriod, 289
- operator(), 289
- storeOffTime, 288
- AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint, 291
- _flightPeriod, 292
- operator(), 292
- storeSegmentBoardingPoint, 291
- AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode, 292
- _flightPeriod, 293
- operator(), 293
- storeSegmentCabinCode, 293
- AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint, 294
- _flightPeriod, 295
- operator(), 295
- storeSegmentOffPoint, 294
- AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity, 295
- _flightPeriod, 296
- operator(), 296
- storeSegmentSpecificity, 296

- airsched::SearchString_T, 225
 - _airlineList, 226
 - _dateList, 226
 - _passengerList, 226
 - _placeList, 226
 - _tmpAirline, 227
 - _tmpDate, 227
 - _tmpPassenger, 227
 - _tmpPlace, 226
 - display, 226
 - SearchString_T, 226
- airsched::SearchStringParser, 227
 - _searchString, 228
 - SearchStringParser, 228
- airsched::SearchStringParser::definition, 164
 - airline_code, 166
 - airline_name, 166
 - date, 166
 - dates, 166
 - day, 166
 - definition, 165
 - month, 166
 - passenger_adult_type, 167
 - passenger_child_type, 167
 - passenger_number, 167
 - passenger_pet_type, 167
 - passenger_type, 167
 - passengers, 166
 - place_element, 166
 - places, 165
 - preferred_airlines, 166
 - search_string, 165
 - start, 165
 - year, 166
- AIRSCHEd::SegmentCabinStruct, 228
 - _cabinCode, 229
 - _classes, 229
 - _fareFamilies, 229
 - _itFamilyCode, 229
 - describe, 229
 - fill, 229
- AIRSCHEd::SegmentDateNotFoundException, 230
 - SegmentDateNotFoundException, 230
- AIRSCHEd::SegmentPathGenerator, 231
 - createSegmentPathNetwork, 231
- AIRSCHEd::SegmentPathPeriod, 231
 - ~SegmentPathPeriod, 233
 - _holderMap, 238
 - _key, 238
 - _parent, 238
 - boost::serialization::access, 237
 - checkCircle, 236
 - connectWithAnotherSegment, 235
 - describeKey, 237
 - fromStream, 237
 - getBoardingDateOffsetList, 234
 - getBoardingTime, 235
 - getDeparturePeriod, 234
 - getDestination, 235
 - getElapsedTime, 234
 - getFirstSegmentPeriod, 235
 - getHolderMap, 235
 - getKey, 233
 - getLastSegmentPeriod, 235
 - getNbOfAirlines, 234
 - getNbOfSegments, 234
 - getParent, 233
 - isAirlineFlown, 236
 - isDepartureDateValid, 236
 - Key_T, 233
 - SegmentPathPeriod, 233
 - serialize, 237
 - stdair::FacBom, 237
 - stdair::FacBomManager, 237
 - toStream, 236
 - toString, 237
- AIRSCHEd::SegmentPathPeriodKey, 238
 - ~SegmentPathPeriodKey, 240
 - boost::serialization::access, 243
 - fromStream, 242
 - getBoardingDateOffsetList, 240
 - getBoardingTime, 241
 - getElapsedTime, 241
 - getNbOfAirlines, 241
 - getNbOfSegments, 240
 - getPeriod, 240
 - isValid, 242
 - SegmentPathPeriodKey, 240
 - serialize, 243
 - setBoardingDateOffsetList, 241
 - setBoardingTime, 242
 - setElapsedTime, 241
 - setNbOfAirlines, 241
 - setPeriod, 241
 - toStream, 242
 - toString, 242
- AIRSCHEd::SegmentPathProvider, 243
 - AIRSCHEd_Service, 243
- AIRSCHEd::SegmentPeriodHelper, 244

- fill, [244](#)
- AIRSCHEd::SegmentStruct, [245](#)
 - _boardingDate, [246](#)
 - _boardingPoint, [246](#)
 - _boardingTime, [246](#)
 - _cabinList, [246](#)
 - _elapsed, [246](#)
 - _offDate, [246](#)
 - _offPoint, [246](#)
 - _offTime, [246](#)
 - describe, [245](#)
 - fill, [245](#)
- AIRSCHEd::ServiceAbstract, [247](#)
 - ~ServiceAbstract, [247](#)
 - fromStream, [248](#)
 - ServiceAbstract, [247](#)
 - toStream, [248](#)
- AIRSCHEd::Simulator, [248](#)
 - simulate, [249](#)
- airsched::store_adult_passenger_type, [249](#)
 - _searchString, [250](#)
 - operator(), [250](#)
 - store_adult_passenger_type, [250](#)
- airsched::store_airline_code, [250](#)
 - _searchString, [251](#)
 - operator(), [251](#)
 - store_airline_code, [251](#)
- airsched::store_airline_name, [251](#)
 - _searchString, [252](#)
 - operator(), [252](#)
 - store_airline_name, [252](#)
- airsched::store_airline_sign, [252](#)
 - _searchString, [253](#)
 - operator(), [253](#)
 - store_airline_sign, [253](#)
- airsched::store_child_passenger_type, [253](#)
 - _searchString, [254](#)
 - operator(), [254](#)
 - store_child_passenger_type, [254](#)
- airsched::store_date, [254](#)
 - _searchString, [255](#)
 - operator(), [255](#)
 - store_date, [255](#)
- airsched::store_passenger_number, [255](#)
 - _searchString, [256](#)
 - operator(), [256](#)
 - store_passenger_number, [256](#)
- airsched::store_pet_passenger_type, [256](#)
 - _searchString, [257](#)
 - operator(), [257](#)
 - store_pet_passenger_type, [257](#)
- airsched::store_place_element, [257](#)
 - _searchString, [258](#)
 - operator(), [258](#)
 - store_place_element, [258](#)
- AIRSCHEd::TravelSolutionParser, [299](#)
 - parseInputFileAndBuildBom, [299](#)
- AIRSCHEd_Service
 - AIRSCHEd::AIRSCHEd_Service, [151](#), [152](#)
 - AIRSCHEd::AIRSCHEd_ServiceContext, [155](#)
 - AIRSCHEd::SegmentPathProvider, [243](#)
- AIRSCHEd_ServicePtr_T
 - AIRSCHEd, [137](#)
- alpha_cap_set_p
 - AIRSCHEd::OnDParserHelper, [142](#)
- BookingRequestParser.cpp
 - BOOST_SPIRIT_DEBUG, [318](#)
 - char_t, [318](#)
 - iterator_t, [318](#)
 - rule_t, [318](#)
 - scanner_t, [318](#)
- boost, [148](#)
- boost::serialization, [148](#)
- boost::serialization::access
 - AIRSCHEd::OriginDestinationSet, [206](#)
 - AIRSCHEd::OriginDestinationSetKey, [209](#)
 - AIRSCHEd::ReachableUniverse, [220](#)
 - AIRSCHEd::ReachableUniverseKey, [223](#)
 - AIRSCHEd::SegmentPathPeriod, [237](#)
 - AIRSCHEd::SegmentPathPeriodKey, [243](#)
- boost::spirit::classic::grammar, [188](#)
- BOOST_SPIRIT_DEBUG
 - BookingRequestParser.cpp, [318](#)
- bounded1_4_p_t
 - AIRSCHEd, [139](#)
- bounded2_p_t
 - AIRSCHEd, [139](#)
- bounded4_p_t
 - AIRSCHEd, [139](#)
- buildSampleBom
 - AIRSCHEd::AIRSCHEd_Service, [153](#)
- buildSegmentPathList
 - AIRSCHEd::AIRSCHEd_Service, [153](#)
- buildSegments
 - AIRSCHEd::FlightPeriodStruct, [184](#)

- cabin_code_p
 - AIRSCHED::ScheduleParserHelper, [146](#)
- char_t
 - AIRSCHED, [137](#)
 - BookingRequestParser.cpp, [318](#)
- checkCircle
 - AIRSCHED::SegmentPathPeriod, [236](#)
- CHILD
 - airsched::Passenger_T, [214](#)
- chset_t
 - AIRSCHED, [138](#)
- class_code_list_p
 - AIRSCHED::ScheduleParserHelper, [147](#)
- class_code_p
 - AIRSCHED::OnDParserHelper, [143](#)
- clean
 - AIRSCHED::FacServiceAbstract, [176](#)
- connectWithAnotherSegment
 - AIRSCHED::SegmentPathPeriod, [235](#)
- CppUnit::TestFixture, [298](#)
- CPPUNIT_TEST_SUITE_REGISTRATION
 - AirlineScheduleTestSuite.hpp, [462](#)
- create
 - AIRSCHED::FacAIRSCHEDServiceContext, [174](#)
- createSegmentPathNetwork
 - AIRSCHED::SegmentPathGenerator, [231](#)
- createStringFromWordList
 - airsched.cpp, [308](#)
- csvDisplay
 - AIRSCHED::AIRSCHED_Service, [154](#)
 - AIRSCHED::BomDisplay, [156](#), [157](#)
- date
 - AIRSCHED::OnDParserHelper::OnDParser::definition, [169](#)
 - AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, [162](#)
 - airsched::SearchStringParser::definition, [166](#)
- date_offset
 - AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, [162](#)
- Date_T
 - airsched::Date_T, [159](#)
- DateList_T
 - airsched, [133](#)
- DateOffsetList_T
 - AIRSCHED, [140](#)
- dates
 - airsched::SearchStringParser::definition, [166](#)
- day
 - airsched::SearchStringParser::definition, [166](#)
- day_p
 - AIRSCHED::OnDParserHelper, [143](#)
 - AIRSCHED::ScheduleParserHelper, [146](#)
- DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
 - AIRSCHED, [141](#)
- definition
 - AIRSCHED::OnDParserHelper::OnDParser::definition, [168](#)
 - AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, [161](#)
 - airsched::SearchStringParser::definition, [165](#)
- describe
 - AIRSCHED::FareFamilyStruct, [177](#)
 - AIRSCHED::FlightPeriodStruct, [183](#)
 - AIRSCHED::LegCabinStruct, [190](#)
 - AIRSCHED::LegStruct, [192](#)
 - AIRSCHED::OnDPeriodStruct, [200](#)
 - AIRSCHED::SegmentCabinStruct, [229](#)
 - AIRSCHED::SegmentStruct, [245](#)
- describeKey
 - AIRSCHED::OriginDestinationSet, [205](#)
 - AIRSCHED::ReachableUniverse, [219](#)
 - AIRSCHED::SegmentPathPeriod, [237](#)
- describeTSKey
 - AIRSCHED::OnDPeriodStruct, [200](#)
- display
 - airsched::Airline_T, [149](#)
 - airsched::Date_T, [159](#)
 - airsched::Passenger_T, [214](#)
 - airsched::Place_T, [216](#)
 - airsched::SearchString_T, [226](#)
- doc/local/authors.doc, [458](#)
- doc/local/codingrules.doc, [458](#)
- doc/local/copyright.doc, [458](#)
- doc/local/documentation.doc, [458](#)
- doc/local/features.doc, [459](#)
- doc/local/help_wanted.doc, [459](#)
- doc/local/howto_release.doc, [459](#)
- doc/local/index.doc, [459](#)
- doc/local/installation.doc, [459](#)
- doc/local/linking.doc, [459](#)
- doc/local/test.doc, [459](#)

- doc/local/users_guide.doc, [459](#)
- doc/local/verification.doc, [459](#)
- doc/tutorial/tutorial.doc, [459](#)
- doEndFlight
 - AIRSCHEd::ScheduleParserHelper::doEndFlight, [161](#)
 - [170](#)
- doEndOnD
 - AIRSCHEd::OnDParserHelper::doEndOnD, [162](#)
 - [172](#)
- dow
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, [162](#)
 - FlightPeriodFileParser
- dow_p
 - AIRSCHEd::ScheduleParserHelper, [146](#)
 - AIRSCHEd::FlightPeriodFileParser, [179](#)
 - AIRSCHEd::InventoryGenerator, [189](#)
 - FlightPeriodParser
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser, [181](#)
 - FlightPeriodStruct
 - AIRSCHEd::FlightPeriodStruct, [183](#)
 - fromStream
 - AIRSCHEd::OriginDestinationSet, [205](#)
 - AIRSCHEd::OriginDestinationSetKey, [208](#)
 - AIRSCHEd::ReachableUniverse, [219](#)
 - AIRSCHEd::ReachableUniverseKey, [223](#)
 - AIRSCHEd::SegmentPathPeriod, [237](#)
 - AIRSCHEd::SegmentPathPeriodKey, [242](#)
 - AIRSCHEd::ServiceAbstract, [248](#)
- externalMemoryManagement
 - AirlineScheduleTestSuite, [150](#)
- FacAIRSCHEdServiceContext
 - AIRSCHEd::AIRSCHEd_ServiceContext, [155](#)
 - AIRSCHEd::FacAIRSCHEdServiceContext, [173](#)
- FacServiceAbstract
 - AIRSCHEd::FacServiceAbstract, [175](#)
- family_cabin_details
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, [164](#)
- family_code_p
 - AIRSCHEd::ScheduleParserHelper, [147](#)
- FareFamilyStruct
 - AIRSCHEd::FareFamilyStruct, [177](#)
- FareFamilyStructList_T
 - AIRSCHEd, [139](#)
- FFFlightPeriodFileParser
 - AIRSCHEd::InventoryGenerator, [189](#)
- fill
 - AIRSCHEd::LegCabinStruct, [190](#)
 - AIRSCHEd::LegStruct, [192](#)
 - AIRSCHEd::SegmentCabinStruct, [229](#)
 - AIRSCHEd::SegmentPeriodHelper, [244](#)
 - AIRSCHEd::SegmentStruct, [245](#)
 - generateInventories
 - AIRSCHEd::FlightPeriodFileParser, [180](#)
 - AIRSCHEd::ScheduleParser, [225](#)
 - generateOnDPeriods
 - AIRSCHEd::OnDParser, [194](#)
 - AIRSCHEd::OnDPeriodFileParser, [197](#)
- FlagSaver
 - AIRSCHEd::FlagSaver, [178](#)
 - generic_segment
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, [164](#)
 - getBoardingDateOnsetList
 - AIRSCHEd::SegmentPathPeriod, [234](#)
 - AIRSCHEd::SegmentPathPeriodKey, [246](#)
- flight_key
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, [162](#)
- flight_number
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, [162](#)
 - getBoardingPoint

- AIRSCHEd::ReachableUniverseKey, 222
- getBoardingTime
 - AIRSCHEd::SegmentPathPeriod, 235
 - AIRSCHEd::SegmentPathPeriodKey, 241
- getDate
 - airsched::Date_T, 159
 - AIRSCHEd::FlightPeriodStruct, 183
 - AIRSCHEd::OnDPeriodStruct, 199
- getDeparturePeriod
 - AIRSCHEd::SegmentPathPeriod, 234
- getDestination
 - AIRSCHEd::OriginDestinationSet, 204
 - AIRSCHEd::SegmentPathPeriod, 235
- getElapsedTime
 - AIRSCHEd::SegmentPathPeriod, 234
 - AIRSCHEd::SegmentPathPeriodKey, 241
- getFirstAirlineCode
 - AIRSCHEd::OnDPeriodStruct, 199
- getFirstSegmentPeriod
 - AIRSCHEd::SegmentPathPeriod, 235
- getHolderMap
 - AIRSCHEd::OriginDestinationSet, 205
 - AIRSCHEd::ReachableUniverse, 218
 - AIRSCHEd::SegmentPathPeriod, 235
- getKey
 - AIRSCHEd::OriginDestinationSet, 204
 - AIRSCHEd::ReachableUniverse, 218
 - AIRSCHEd::SegmentPathPeriod, 233
- getLastSegmentPeriod
 - AIRSCHEd::SegmentPathPeriod, 235
- getNbOfAirlines
 - AIRSCHEd::SegmentPathPeriod, 234
 - AIRSCHEd::SegmentPathPeriodKey, 241
- getNbOfSegments
 - AIRSCHEd::SegmentPathPeriod, 234
 - AIRSCHEd::SegmentPathPeriodKey, 240
- getOffPoint
 - AIRSCHEd::OriginDestinationSetKey, 208
- getOrigin
 - AIRSCHEd::ReachableUniverse, 218
- getParent
 - AIRSCHEd::OriginDestinationSet, 205
 - AIRSCHEd::ReachableUniverse, 218
 - AIRSCHEd::SegmentPathPeriod, 233
- getPeriod
 - AIRSCHEd::SegmentPathPeriodKey, 240
 - getSegmentPathPeriodListList
 - AIRSCHEd::ReachableUniverse, 218
 - getTime
 - AIRSCHEd::FlightPeriodStruct, 183
 - AIRSCHEd::OnDPeriodStruct, 200
 - hours_p
 - AIRSCHEd::OnDParserHelper, 143
 - AIRSCHEd::ScheduleParserHelper, 146
 - instance
 - AIRSCHEd::FacAIRSCHEdServiceContext, 174
 - int1_p
 - airsched, 134
 - AIRSCHEd::ScheduleParserHelper, 147
 - int1_p_t
 - AIRSCHEd, 138
 - isAirlineFlown
 - AIRSCHEd::SegmentPathPeriod, 236
 - isDepartureDateValid
 - AIRSCHEd::SegmentPathPeriod, 236
 - isValid
 - AIRSCHEd::SegmentPathPeriodKey, 242
 - iterator_t
 - AIRSCHEd, 138
 - BookingRequestParser.cpp, 318
 - jsonExport
 - AIRSCHEd::AIRSCHEd_Service, 154
 - K_AIRSCHEd_DEFAULT_BOOKING_REQUEST
 - airsched.cpp, 308
 - K_AIRSCHEd_DEFAULT_BOOKING_REQUEST_MODE
 - airsched.cpp, 309
 - K_AIRSCHEd_DEFAULT_BUILT_IN_INPUT
 - airsched.cpp, 309
 - K_AIRSCHEd_DEFAULT_INPUT_FILENAME
 - airsched.cpp, 308
 - K_AIRSCHEd_DEFAULT_LOG_FILENAME
 - airsched.cpp, 308
 - K_AIRSCHEd_EARLY_RETURN_STATUS
 - airsched.cpp, 309
 - Key_T
 - AIRSCHEd::OriginDestinationSet, 204
 - AIRSCHEd::ReachableUniverse, 217

AIRSCHED::SegmentPathPeriod, 233
 LAST_VALUE
 airsched::Passenger_T, 214
 leg
 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, 163
 leg_cabin_details
 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, 163
 leg_details
 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, 163
 leg_key
 AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition, 163
 LegCabinStructList_T
 AIRSCHED, 139
 LegStruct
 AIRSCHED::LegStruct, 192
 LegStructList_T
 AIRSCHED, 139
 main
 airsched.cpp, 309
 minutes_p
 AIRSCHED::OnDParserHelper, 143
 AIRSCHED::ScheduleParserHelper, 146
 month
 airsched::SearchStringParser::definition, 166
 month_p
 AIRSCHED::OnDParserHelper, 143
 AIRSCHED::ScheduleParserHelper, 145
 ond
 AIRSCHED::OnDParserHelper::OnDParser::definition, 168
 ond_end
 AIRSCHED::OnDParserHelper::OnDParser::definition, 169
 ond_key
 AIRSCHED::OnDParserHelper::OnDParser::definition, 169
 ond_list
 AIRSCHED::OnDParserHelper::OnDParser::definition, 168
 OnDInputFileNotFoundException
 AIRSCHED::OnDInputFileNotFoundException, 194
 AIRSCHED::OnDParser
 AIRSCHED::OnDParserHelper::OnDParser, 196
 AIRSCHED::OnDPeriodGenerator, 198
 OnDParserHelper::doEndOnD
 AIRSCHED::OnDPeriodGenerator, 198
 OnDPeriodFileParser
 AIRSCHED::OnDPeriodFileParser, 197
 AIRSCHED::OnDPeriodGenerator, 198
 OnDPeriodStruct
 AIRSCHED::OnDPeriodStruct, 199
 AIRSCHED::ScheduleParserHelper::definition, operator<, 308
 airsched.cpp, 308
 ServiceAbstract.hpp, 457
 AIRSCHED::ScheduleParserHelper::definition, operator>, 457
 ServiceAbstract.hpp, 457
 operator()
 AIRSCHED::OnDParserHelper::doEndOnD, 172
 AIRSCHED::OnDParserHelper::storeAirlineCode, 259
 AIRSCHED::OnDParserHelper::storeClassCode, 265
 AIRSCHED::OnDParserHelper::storeDateRangeEnd, 269
 AIRSCHED::OnDParserHelper::storeDateRangeStart, 272
 AIRSCHED::OnDParserHelper::storeDestination, 274
 AIRSCHED::OnDParserHelper::storeEndRangeTime, 278
 AIRSCHED::OnDParserHelper::storeOrigin, 290
 AIRSCHED::OnDParserHelper::storeStartRangeTime, 298
 AIRSCHED::ScheduleParserHelper::doEndFlight, 170
 AIRSCHED::ScheduleParserHelper::storeAirlineCode, 260
 AIRSCHED::ScheduleParserHelper::storeBoardingTime, 262
 AIRSCHED::ScheduleParserHelper::storeCapacity, 263
 AIRSCHED::ScheduleParserHelper::storeClasses, 266
 AIRSCHED::ScheduleParserHelper::storeDateRangeEnd, 268
 AIRSCHED::ScheduleParserHelper::storeDateRangeStart, 271

AIRSCHED::ScheduleParserHelper::storeOriginDestinationSetKey::serialize< ba::text_
 275 iarchive >
 AIRSCHED::ScheduleParserHelper::storeElapseTime, 141
 277 OriginDestinationSetKey::serialize< ba::text_
 AIRSCHED::ScheduleParserHelper::storeFamilyCode, 141
 279 iarchive >
 AIRSCHED::ScheduleParserHelper::storeFlightDate, 141
 281 OriginDestinationSetList_T
 AIRSCHED::ScheduleParserHelper::storeFlightDate, 139
 283 OriginDestinationSetMap_T
 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint,
 284 parseAndLoad
 AIRSCHED::ScheduleParserHelper::storeLegClassCode, AIRSCHED::AIRSCHED_Service, 152,
 286 153
 AIRSCHED::ScheduleParserHelper::storeParseBookingRequest
 287 airsched, 133
 AIRSCHED::ScheduleParserHelper::storeOffline, 309
 289 parseInputFileAndBuildBom
 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint, AIRSCHED::TravelSolutionParser, 299
 292 ParserSemanticAction
 AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode, AIRSCHED::ODP
 293 ParserSemanticAction, 210
 AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint, AIRSCHED::ScheduleParserHelper::ParserSemanticAction,
 295 212
 AIRSCHED::ScheduleParserHelper::storepassengeradulttype,
 296 airsched::SearchStringParser::definition,
 airsched::store_adult_passenger_type, 167
 250 passenger_child_type
 airsched::store_airline_code, 251 airsched::SearchStringParser::definition,
 airsched::store_airline_name, 252 167
 airsched::store_airline_sign, 253 passenger_number
 airsched::store_child_passenger_type, airsched::SearchStringParser::definition,
 254 167
 airsched::store_date, 255 passenger_pet_type
 airsched::store_passenger_number, 256 airsched::SearchStringParser::definition,
 airsched::store_pet_passenger_type, 167
 257 Passenger_T
 airsched::store_place_element, 258 airsched::Passenger_T, 214
 OriginDestinationSet
 AIRSCHED::OriginDestinationSet, 204
 OriginDestinationSet::serialize< ba::text_
 iarchive >
 AIRSCHED, 141
 OriginDestinationSet::serialize< ba::text_
 oarchive >
 AIRSCHED, 141
 OriginDestinationSetKey
 AIRSCHED::OriginDestinationSetKey,
 208 PET
 airsched::Passenger_T, 214

- place_element
 - airsched::SearchStringParser::definition, 166
- Place_T
 - airsched::Place_T, 215
- PlaceList_T
 - airsched, 133
- places
 - airsched::SearchStringParser::definition, 165
- preferred_airlines
 - airsched::SearchStringParser::definition, 166
- ReachableUniverse
 - AIRSCHEd::ReachableUniverse, 218
- ReachableUniverse::serialize< ba::text_iarchive >
 - AIRSCHEd, 141
- ReachableUniverse::serialize< ba::text_oarchive >
 - AIRSCHEd, 141
- ReachableUniverseKey
 - AIRSCHEd::ReachableUniverseKey, 222
- ReachableUniverseKey::serialize< ba::text_iarchive >
 - AIRSCHEd, 141
- ReachableUniverseKey::serialize< ba::text_oarchive >
 - AIRSCHEd, 141
- ReachableUniverseList_T
 - AIRSCHEd, 140
- ReachableUniverseMap_T
 - AIRSCHEd, 140
- readConfiguration
 - airsched.cpp, 308
- repeat_p_t
 - AIRSCHEd, 138
- rule_t
 - AIRSCHEd, 138
 - BookingRequestParser.cpp, 318
- scanner_t
 - AIRSCHEd, 138
 - BookingRequestParser.cpp, 318
- ScheduleInputFileNotFoundException
 - AIRSCHEd::ScheduleInputFileNotFoundException, 224
- ScheduleParser
 - AIRSCHEd::InventoryGenerator, 189
- ScheduleParserHelper::doEndFlight
 - AIRSCHEd::InventoryGenerator, 189
- scheduleParsing
 - AirlineScheduleTestSuite, 150
- search_string
 - airsched::SearchStringParser::definition, 165
- SearchString_T
 - airsched::SearchString_T, 226
- SearchStringParser
 - airsched::SearchStringParser, 228
- seconds_p
 - AIRSCHEd::OnDParserHelper, 143
 - AIRSCHEd::ScheduleParserHelper, 146
- segment
 - AIRSCHEd::OnDParserHelper::OnDParser::definition, 169
- segment_cabin_details
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, 163
- segment_key
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, 163
- segment_section
 - AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, 163
- SegmentCabinStructList_T
 - AIRSCHEd, 140
- SegmentDateNotFoundException
 - AIRSCHEd::SegmentDateNotFoundException, 230
- SegmentPathGenerator
 - AIRSCHEd::ReachableUniverse, 220
- SegmentPathPeriod
 - AIRSCHEd::SegmentPathPeriod, 233
- SegmentPathPeriodKey
 - AIRSCHEd::SegmentPathPeriodKey, 240
- SegmentPathPeriodLightList_T
 - AIRSCHEd, 140
- SegmentPathPeriodList_T
 - AIRSCHEd, 140
- SegmentPathPeriodListList_T
 - AIRSCHEd, 140
- SegmentPathPeriodMultimap_T
 - AIRSCHEd, 140
- SegmentStructList_T
 - AIRSCHEd, 141
- serialize
 - AIRSCHEd::OriginDestinationSet, 206

- AIRSCHEd::OriginDestinationSetKey, 209
- AIRSCHEd::ReachableUniverse, 219
- AIRSCHEd::ReachableUniverseKey, 228
- AIRSCHEd::SegmentPathPeriod, 237
- AIRSCHEd::SegmentPathPeriodKey, 243
- ServiceAbstract
 - AIRSCHEd::ServiceAbstract, 247
- ServiceAbstract.hpp
 - operator<=, 457
 - operator>, 457
- ServicePool_T
 - AIRSCHEd::FacServiceAbstract, 175
- setBoardingDateOffsetList
 - AIRSCHEd::SegmentPathPeriodKey, 241
- setBoardingTime
 - AIRSCHEd::SegmentPathPeriodKey, 242
- setElapsedTime
 - AIRSCHEd::SegmentPathPeriodKey, 241
- setNbOfAirlines
 - AIRSCHEd::SegmentPathPeriodKey, 241
- setPeriod
 - AIRSCHEd::SegmentPathPeriodKey, 241
- simulate
 - AIRSCHEd::AIRSCHEd_Service, 154
 - AIRSCHEd::Simulator, 249
- specific_segment_list
 - AIRSCHEd::ScheduleParserHelper::FlightPeriod::definition, 164
- start
 - AIRSCHEd::OnDParserHelper::OnDParser::definition, 168
 - AIRSCHEd::ScheduleParserHelper::FlightPeriod::definition, 161
 - airsched::SearchStringParser::definition, 165
- stdair, 148
- stdair::BomAbstract, 156
- stdair::CmdAbstract, 157
- stdair::FacBom
 - AIRSCHEd::OriginDestinationSet, 206
 - AIRSCHEd::ReachableUniverse, 220
 - AIRSCHEd::SegmentPathPeriod, 237
- stdair::FacBomManager
 - AIRSCHEd::OriginDestinationSet, 206
 - AIRSCHEd::ReachableUniverse, 220
 - AIRSCHEd::SegmentPathPeriod, 237
- stdair::FacServiceAbstract, 176
- stdair::FileNotFoundException, 178
- stdair::KeyAbstract, 189
- stdair::ParserException, 209
- stdair::ServiceAbstract, 247
- stdair::StructAbstract, 298
- store_adult_passenger_type
 - airsched::store_adult_passenger_type, 250
- store_airline_code
 - airsched::store_airline_code, 251
- store_airline_name
 - airsched::store_airline_name, 252
- store_airline_sign
 - airsched::store_airline_sign, 253
- store_child_passenger_type
 - airsched::store_child_passenger_type, 254
- store_date
 - airsched::store_date, 255
- store_passenger_number
 - airsched::store_passenger_number, 256
- store_pet_passenger_type
 - airsched::store_pet_passenger_type, 257
- store_place_element
 - airsched::store_place_element, 258
- storeAirlineCode
 - AIRSCHEd::OnDParserHelper::storeAirlineCode, 259
 - AIRSCHEd::ScheduleParserHelper::storeAirlineCode, 260
- storeBoardingTime
 - AIRSCHEd::ScheduleParserHelper::storeBoardingTime, 262
- storeCapacity
 - AIRSCHEd::ScheduleParserHelper::storeCapacity, 263
- storeClassCode
 - AIRSCHEd::OnDParserHelper::storeClassCode, 265
- storeClasses
 - AIRSCHEd::ScheduleParserHelper::storeClasses, 266
- storeDateRangeEnd
 - AIRSCHEd::OnDParserHelper::storeDateRangeEnd, 269

- AIRSCHEd::ScheduleParserHelper::storeDateRangeEnd, 268
- AIRSCHEd::ScheduleParserHelper::storeDateRangeStart, 272
- AIRSCHEd::ScheduleParserHelper::storeDateRangeTime, 271
- storeDestination, 273
 - AIRSCHEd::OnDParserHelper::storeDestination, 273
- storeDow, 275
 - AIRSCHEd::ScheduleParserHelper::storeDow, 275
- storeElapsedTime, 276
 - AIRSCHEd::ScheduleParserHelper::storeElapsedTime, 276
- storeEndRangeTime, 278
 - AIRSCHEd::OnDParserHelper::storeEndRangeTime, 278
- storeFamilyCode, 279
 - AIRSCHEd::ScheduleParserHelper::storeFamilyCode, 279
- storeFClasses, 281
 - AIRSCHEd::ScheduleParserHelper::storeFClasses, 281
- storeFlightNumber, 282
 - AIRSCHEd::ScheduleParserHelper::storeFlightNumber, 282
- storeLegBoardingPoint, 284
 - AIRSCHEd::ScheduleParserHelper::storeLegBoardingPoint, 284
- storeLegCabinCode, 285
 - AIRSCHEd::ScheduleParserHelper::storeLegCabinCode, 285
- storeLegOffPoint, 287
 - AIRSCHEd::ScheduleParserHelper::storeLegOffPoint, 287
- storeOffTime, 288
 - AIRSCHEd::ScheduleParserHelper::storeOffTime, 288
- storeOrigin, 290
 - AIRSCHEd::OnDParserHelper::storeOrigin, 290
- storeSegmentBoardingPoint, 291
 - AIRSCHEd::ScheduleParserHelper::storeSegmentBoardingPoint, 291
- storeSegmentCabinCode, 293
 - AIRSCHEd::ScheduleParserHelper::storeSegmentCabinCode, 293
- storeSegmentOffPoint, 294
 - AIRSCHEd::ScheduleParserHelper::storeSegmentOffPoint, 294
- storeSegmentSpecificity, 296
 - AIRSCHEd::ScheduleParserHelper::storeSegmentSpecificity, 296
- storeStartRangeTime, 297
 - AIRSCHEd::OnDParserHelper::storeStartRangeTime, 297
- test/ Directory Reference, 132
- test/airsched/ Directory Reference, 129
- test/airsched/ AirlineScheduleTestSuite.cpp, 459
- test/airsched/ AirlineScheduleTestSuite.hpp, 461, 462
- AIRSCHEd::OnDParserHelper::OnDParser::definition, 169
- AIRSCHEd::ScheduleParserHelper::FlightPeriodParser::definition, 162
- toSummary, 205
- AIRSCHEd::OriginDestinationSet, 205
- AIRSCHEd::OriginDestinationSetKey, 208
- AIRSCHEd::ReachableUniverse, 219
- AIRSCHEd::ReachableUniverseKey, 222
- AIRSCHEd::SegmentPathPeriod, 236
- AIRSCHEd::SegmentPathPeriodKey, 242
- AIRSCHEd::ServiceAbstract, 248
- toString, 205
- AIRSCHEd::OriginDestinationSet, 205
- AIRSCHEd::OriginDestinationSetKey, 209
- AIRSCHEd::ReachableUniverse, 219
- AIRSCHEd::ReachableUniverseKey, 223
- AIRSCHEd::SegmentPathPeriod, 237
- AIRSCHEd::SegmentPathPeriodKey, 242
- uint1_2_p, 134
- airsched, 134
- uint1_4_p, 134
- airsched, 134
- AIRSCHEd::OnDParserHelper, 144
- AIRSCHEd::ScheduleParserHelper, 147
- uint1_4_p_t, 138
- AIRSCHEd, 138
- uint1_p, 134
- airsched, 134

uint2_4_p
 airsched, [134](#)

uint2_p
 airsched, [134](#)
 AIRSCHED::OnDParserHelper, [144](#)
 AIRSCHED::ScheduleParserHelper, [147](#)

uint2_p_t
 AIRSCHED, [138](#)

uint4_p
 airsched, [134](#)
 AIRSCHED::OnDParserHelper, [144](#)
 AIRSCHED::ScheduleParserHelper, [147](#)

uint4_p_t
 AIRSCHED, [138](#)

WordList_T
 airsched.cpp, [308](#)

year
 airsched::SearchStringParser::definition,
 [166](#)

year_p
 AIRSCHED::OnDParserHelper, [143](#)
 AIRSCHED::ScheduleParserHelper, [145](#)