AirSched

0.1.4

Generated by Doxygen 1.8.1.2

Thu Aug 16 2012 08:13:08

Contents

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
- SOCI (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 "0.1.4"
#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-0.1.4"
#define MANDIR "/usr/share/man"
#define INFODIR "/usr/share/info"
#define HTMLDIR "/usr/share/doc/airsched-0.1.4/html"
#define PDFDIR "/usr/share/doc/airsched-0.1.4/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

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.

5.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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 Release branch maintenance

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

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

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

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

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

```
vi NEWS
vi ChangeLog
vi 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 the 0.5.0 version of AirSched."
git push
```

9.5 Create distribution packages

Create the distribution packages using the following command:

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

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

9.6 Upload the HTML documentation to SourceForge

In order to update the Web site files, either:

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

  ```
  cd ~/dev/sim/airschedgit/build
  git checkout releases
  rsync -aiv ${INSTALL_BASEDIR}/airsched-0.5.0/share/doc/airsched-0.5.0/html/ \
   your_sf_user,air-sched@web.sourceforge.net:htdocs/
  ```

  where `-aiv` options mean:

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

- or use the `SourceForge Shell service`.

9.7 Generate the RPM packages

Optionally, generate the RPM package (for instance, for `Fedora`/`RedHat`):

```
cd ~/dev/sim/airschedgit/build
git checkout releases
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
cd ~/dev/packages
rpmlint -i SPECS/airsched.spec SRPMS/airsched-0.5.0-1.fc16.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:

```
cd ~/dev/sim/airschedgit/build
git checkout releases
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 Make a new post

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

9.12 Send an email on the announcement mailing-list

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

10 Installation

10.1 Table of Contents

- 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')
- `SOCI` - C++ database client library wrapper, version 3.0.0 or later (check version with `'soci-config` –version')

Optionally, you might need a few additional programs: `Doxygen`, `LaTeX`, `Dvips` and `Ghostscript`, to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of 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 <tt>'CMakeLists.txt'</tt> is used to create the \c 'Makefile'
```

files.

The simplest way to compile this package is:

1. `'cd'` to the directory containing the package's source code and type `'./cmake ..'` to configure the package for your system. Running `'cmake'` is generally fast. While running, it prints some messages telling which features it is checking for.

2. Type `'make'` to compile the package.

3. Optionally, type `'make check'to run any self-tests that come with the package, generally using the just-built uninstalled binaries.`

4. `Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.`

5. `You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.`

6. `Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.`

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=P- REFIX', 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=P- REFIX' 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=P-REFIX' 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 ouput 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:ST
-- 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: 6100bb1479e9c72f807a60067138dfe1b71cbec7 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 ................... : airsched
-- PACKAGE_PRETTY_NAME ............ : AirSched
-- PACKAGE ........................ : airsched
-- 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 ............... : airsched
-- Libraries to build/install ..... : airschedlib
-- Binaries to build/install ...... : airsched
-- Modules to test ................ : airsched
-- Binaries to test ............... : AirlineScheduleTestSuitetst
--
-- * Module ....................... : airsched
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : airschedlib
--   + Executables to build/install : airsched
--   + 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/samples
-- INSTALL_DOC .................... : ON
--
-- ---------------------------------
-- ---   Packaging Configuration  ---
-- ---------------------------------
-- CPACK_PACKAGE_CONTACT .......... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ........... : Denis Arnaud
-- CPACK_PACKAGE_VERSION .......... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/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;filesystem;unit_
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_rege
--
-- * 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
--   - SOCIMYSQL_INCLUDE_DIR ...... : /usr/include/soci
--   - SOCI_LIBRARIES ............. : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ........ : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ............. : 0.38.0
--   - STDAIR_BINARY_DIRS ......... : /home/user/dev/deliveries/stdair-0.38.0/bin
--   - STDAIR_EXECUTABLES ......... : stdair
--   - STDAIR_LIBRARY_DIRS ........ : /home/user/dev/deliveries/stdair-0.38.0/lib64
--   - STDAIR_LIBRARIES ........... : stdairlib;stdairuicllib
--   - STDAIR_INCLUDE_DIRS ........ : /home/user/dev/deliveries/stdair-0.38.0/include
--   - STDAIR_SAMPLE_DIR .......... : /home/user/dev/deliveries/stdair-0.38.0/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ============================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/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

11.2 Introdution

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`).

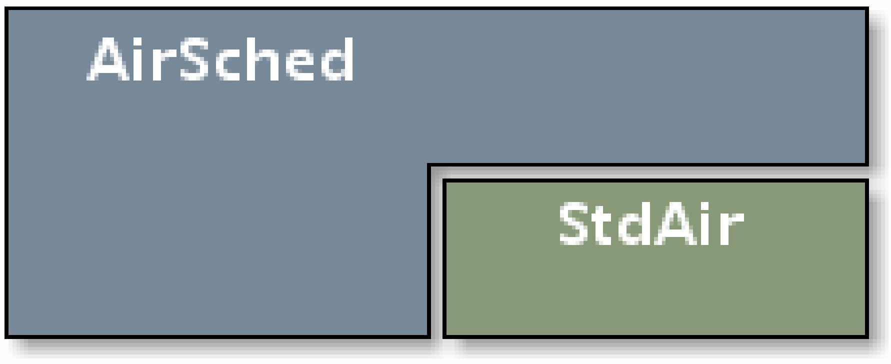


Figure 1: AirSched Dependencies

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 he `'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

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
      ; 20; Y; 300; 0; F; FA; 1; FA; J; JCDI; 1; JCDI; W; WT; 1; WT; Y; YBHKMLSQ; 1;
      YBHKMLSQ;
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
      ; 20; Y; 300; 1; LHR; BKK; F; FA; J; JCDI; W; WT; Y; YBHKMLSQ; BKK; SYD; F; FA;
      J; JCDI; W; WT; Y; YBHKMLSQ; LHR; SYD; F; FA; 1; FA; J; JCDI; 1; JCDI; 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; JCDI; 1; JCDI; 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; JCDI; 1; JCDI; 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; JCDI; 1; JCDI; W; WT; 1; WT; Y;
      YBHKMRVNELSQO; 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; JCDI; 1; JCDI; W; WT; 1; WT; Y;
      YBHKMRVNELSQO; 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; JCDI; 1; JCDI; W; WT; 1; WT; Y;
      YBHKMRVNELSQO; 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 bahave 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`, `JCDI`, `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`, `JCDI`, `WT` or `YBHK-MLSQ`)

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::build-SampleBom()` 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,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-15, SIN-BKK, 2010-Jan-15, 08:20:00, 2010-Jan-15, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 2, 298
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, 0, 0, 0, 2, 298, 0,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, 0, 0, 0, 2, 298, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, Y, 300 (0), 0, 0, 0, 2, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-16
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-16, SIN-BKK, 2010-Jan-16, 08:20:00, 2010-Jan-16, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 1.83244e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-17
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-17, SIN-BKK, 2010-Jan-17, 08:20:00, 2010-Jan-17, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 1.58896e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-18
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-18, SIN-BKK, 2010-Jan-18, 08:20:00, 2010-Jan-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
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,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-19, SIN-BKK, 2010-Jan-19, 08:20:00, 2010-Jan-19, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-20
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-20, SIN-BKK, 2010-Jan-20, 08:20:00, 2010-Jan-20, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-21
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-21, SIN-BKK, 2010-Jan-21, 08:20:00, 2010-Jan-21, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-22
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-22, SIN-BKK, 2010-Jan-22, 08:20:00, 2010-Jan-22, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
```

```
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-23
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-23, SIN-BKK, 2010-Jan-23, 08:20:00, 2010-Jan-23, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 6.64029e-
      319, 0, 300, 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-24
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-24, SIN-BKK, 2010-Jan-24, 08:20:00, 2010-Jan-24, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-25, SIN-BKK, 2010-Jan-25, 08:20:00, 2010-Jan-25, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-26, SIN-BKK, 2010-Jan-26, 08:20:00, 2010-Jan-26, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-27
```

```
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-27, SIN-BKK, 2010-Jan-27, 08:20:00, 2010-Jan-27, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-28, SIN-BKK, 2010-Jan-28, 08:20:00, 2010-Jan-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-29
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-29, SIN-BKK, 2010-Jan-29, 08:20:00, 2010-Jan-29, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-30
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-30, SIN-BKK, 2010-Jan-30, 08:20:00, 2010-Jan-30, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-31
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-31, SIN-BKK, 2010-Jan-31, 08:20:00, 2010-Jan-31, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
```

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

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

```
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-05
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-05, SIN-BKK, 2010-Feb-05, 08:20:00, 2010-Feb-05, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-06
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-06, SIN-BKK, 2010-Feb-06, 08:20:00, 2010-Feb-06, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
```

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

```
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-09
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
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,
      CommSpace, 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,
      Elapsed, 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,
      CommSpace, 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,
      Elapsed, 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,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-12
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-12, SIN-BKK, 2010-Feb-12, 08:20:00, 2010-Feb-12, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
```

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

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

```
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-18
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-18, SIN-BKK, 2010-Feb-18, 08:20:00, 2010-Feb-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-19
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-19, SIN-BKK, 2010-Feb-19, 08:20:00, 2010-Feb-19, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-20
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-20, SIN-BKK, 2010-Feb-20, 08:20:00, 2010-Feb-20, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-21
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-21, SIN-BKK, 2010-Feb-21, 08:20:00, 2010-Feb-21, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
```

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

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

```
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-26, SIN-BKK, 2010-Feb-26, 08:20:00, 2010-Feb-26, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-27
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-27, SIN-BKK, 2010-Feb-27, 08:20:00, 2010-Feb-27, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-28, SIN-BKK, 2010-Feb-28, 08:20:00, 2010-Feb-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-15
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-15, SIN-HND, 2010-Jan-15, 09:20:00, 2010-Jan-15, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 200, 200, 2.082e+121, 5.53287e-48, 5.
      20268e-90, 0, 1.31346e-47, 1.05119e-153, 2.78986e+179, 0, 200, 9, 3.66962e-62, 1
      .0854e-71, 6.74783e-67, 6.9835e-77, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, Y13856, 200 (0), 0, 0, 0, 0, 0 (0)
      , 0, 0, 0, 0, 0, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-16
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-16, SIN-HND, 2010-Jan-16, 09:20:00, 2010-Jan-16, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 2.63638e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-17
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, 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,
      CommSpace, 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,
      Elapsed, 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,
      CommSpace, 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, 0, 200
      , 9, 2.14469e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-19
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-19, SIN-HND, 2010-Jan-19, 09:20:00, 2010-Jan-19, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-20
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-20, SIN-HND, 2010-Jan-20, 09:20:00, 2010-Jan-20, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-21
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-21, SIN-HND, 2010-Jan-21, 09:20:00, 2010-Jan-21, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-22
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-22, SIN-HND, 2010-Jan-22, 09:20:00, 2010-Jan-22, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
```

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

```
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, 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,
      CommSpace, 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,
******************************************
******************************************
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,
      Elapsed, 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,
      CommSpace, 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, 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,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-27, SIN-HND, 2010-Jan-27, 09:20:00, 2010-Jan-27, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-28, SIN-HND, 2010-Jan-28, 09:20:00, 2010-Jan-28, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, 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,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-29, SIN-HND, 2010-Jan-29, 09:20:00, 2010-Jan-29, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-30
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-30, SIN-HND, 2010-Jan-30, 09:20:00, 2010-Jan-30, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
```

```
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-31
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-31, SIN-HND, 2010-Jan-31, 09:20:00, 2010-Jan-31, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-01
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-01, SIN-HND, 2010-Feb-01, 09:20:00, 2010-Feb-01, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-02
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-02, SIN-HND, 2010-Feb-02, 09:20:00, 2010-Feb-02, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-03
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-03, SIN-HND, 2010-Feb-03, 09:20:00, 2010-Feb-03, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-04
******************************************
```

```
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-04, SIN-HND, 2010-Feb-04, 09:20:00, 2010-Feb-04, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-05
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-05, SIN-HND, 2010-Feb-05, 09:20:00, 2010-Feb-05, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-06
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-06, SIN-HND, 2010-Feb-06, 09:20:00, 2010-Feb-06, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-07
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-07, SIN-HND, 2010-Feb-07, 09:20:00, 2010-Feb-07, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-08
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-08, SIN-HND, 2010-Feb-08, 09:20:00, 2010-Feb-08, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
```

```
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-09
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-09, SIN-HND, 2010-Feb-09, 09:20:00, 2010-Feb-09, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-10
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-10, SIN-HND, 2010-Feb-10, 09:20:00, 2010-Feb-10, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-11
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-11, SIN-HND, 2010-Feb-11, 09:20:00, 2010-Feb-11, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-12
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-12, SIN-HND, 2010-Feb-12, 09:20:00, 2010-Feb-12, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-13
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-13, SIN-HND, 2010-Feb-13, 09:20:00, 2010-Feb-13, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-14
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-14, SIN-HND, 2010-Feb-14, 09:20:00, 2010-Feb-14, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-15
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-15, SIN-HND, 2010-Feb-15, 09:20:00, 2010-Feb-15, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 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,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-16, SIN-HND, 2010-Feb-16, 09:20:00, 2010-Feb-16, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
```

```
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-17
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-17, SIN-HND, 2010-Feb-17, 09:20:00, 2010-Feb-17, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-18
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-18, SIN-HND, 2010-Feb-18, 09:20:00, 2010-Feb-18, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-19
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-19, SIN-HND, 2010-Feb-19, 09:20:00, 2010-Feb-19, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-20
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-20, SIN-HND, 2010-Feb-20, 09:20:00, 2010-Feb-20, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-21
```

```
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-21, SIN-HND, 2010-Feb-21, 09:20:00, 2010-Feb-21, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-22
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-22, SIN-HND, 2010-Feb-22, 09:20:00, 2010-Feb-22, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-23
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-23, SIN-HND, 2010-Feb-23, 09:20:00, 2010-Feb-23, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-24
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-24, SIN-HND, 2010-Feb-24, 09:20:00, 2010-Feb-24, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-25, SIN-HND, 2010-Feb-25, 09:20:00, 2010-Feb-25, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
******************************************
******************************************
```

```
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, 0, 0, 0, 0, 200, 0,
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,
      Elapsed, 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,
      CommSpace, 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,
      Elapsed, 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,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
```

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

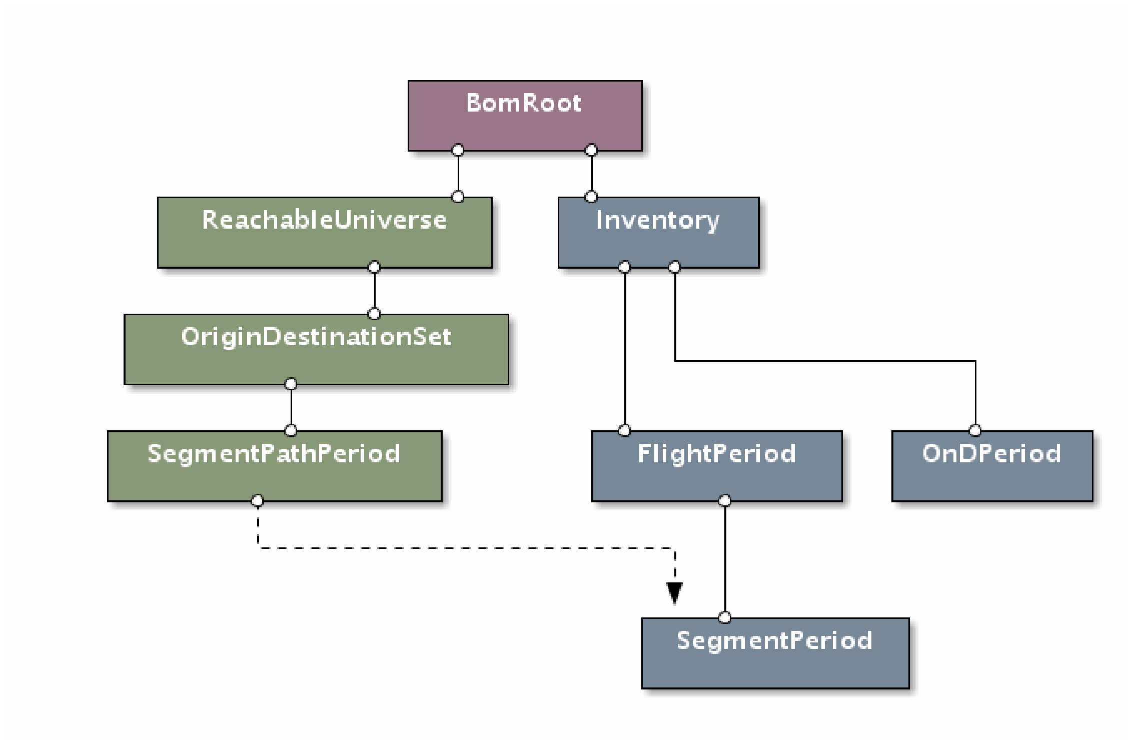


Figure 2: AirSched BOM tree

`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

14.2 Introdution

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 sucessful. 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

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 builde 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 Bristish 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/AIRSCHED_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 Namespace Index

18.1 Namespace List

Here is a list of all namespaces with brief descriptions:

19 Class Index

19.1 Class Hierarchy

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

20 Class Index

20.1 Class List

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

21 File Index

21.1 File List

Here is a list of all files with brief descriptions:

22 Namespace Documentation

22.1 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

22.1.1 Typedef Documentation

22.1.1.1 typedef std::vector<Place_T> airsched::PlaceList_T

List of Place strucutres.

Definition at line 24 of file BookingRequestParser.hpp.

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

List of Date strucutres.

Definition at line 49 of file BookingRequestParser.hpp.

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

List of Airline strucutres.

Definition at line 68 of file BookingRequestParser.hpp.

22.1.1.4 typedef std::vector< Passenger_T > airsched::PassengerList_T

List of Passenger strucutres.

Definition at line 91 of file BookingRequestParser.hpp.

22.1.2 Function Documentation

22.1.2.1 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.

22.1.3 Variable Documentation

22.1.3.1 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.

22.1.3.2 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().

22.1.3.3 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().

22.1.3.4 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().

22.1.3.5 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.

22.1.3.6 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().

22.1.3.7 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.

22.2 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)
- template void SegmentPathPeriod::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentPathPeriod::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentPathPeriodKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentPathPeriodKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

Variables

- const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000

22.2.1 Typedef Documentation

22.2.1.1 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.

22.2.1.2 typedef char AIRSCHED::char_t

Definition at line 31 of file BasParserTypes.hpp.

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

Definition at line 35 of file BasParserTypes.hpp.

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

Definition at line 36 of file BasParserTypes.hpp.

22.2.1.5 typedef boost::spirit::classic::rule<scanner_t> AIRSCHED::rule_t

Definition at line 37 of file BasParserTypes.hpp.

22.2.1.6 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.

22.2.1.7 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.

22.2.1.8 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.

22.2.1.9 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.

22.2.1.10 typedef boost::spirit::classic::chset<char_t> AIRSCHED::chset_t

character set

Definition at line 57 of file BasParserTypes.hpp.

22.2.1.11 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.

22.2.1.12 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.

22.2.1.13 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> AIRSCHED::bounded4_p_t

Definition at line 67 of file BasParserTypes.hpp.

22.2.1.14 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int> AIRSCHED::bounded1_4_p_t

Definition at line 68 of file BasParserTypes.hpp.

22.2.1.15 typedef std::set<stdair::AirportCode_T> AIRSCHED::AirportList_T

Define lists of Airport Codes.

Definition at line 16 of file AirportList.hpp.

22.2.1.16 typedef std::vector<stdair::AirportCode_T> AIRSCHED::AirportOrderedList_T

Definition at line 17 of file AirportList.hpp.

22.2.1.17 typedef std::vector<FareFamilyStruct> AIRSCHED::FareFamilyStructList_T

List of FareFamily-Detail structures.

Definition at line 31 of file FareFamilyStruct.hpp.

22.2.1.18 typedef std::vector<LegCabinStruct> AIRSCHED::LegCabinStructList_T

List of LegCabin-Detail strucutres.

Definition at line 36 of file LegCabinStruct.hpp.

22.2.1.19 typedef std::vector<LegStruct> AIRSCHED::LegStructList_T

List of Leg strucutres.

Definition at line 50 of file LegStruct.hpp.

22.2.1.20 typedef std::list<OriginDestinationSet*> AIRSCHED::OriginDestinationSetList_T

Define the OriginDestinationSet list.

Definition at line 18 of file OriginDestinationSetTypes.hpp.

22.2.1.21 typedef std::map<const stdair::MapKey_T, OriginDestinationSet*> AIRSCHED::OriginDestinationSet-Map_T

Define the OriginDestinationSet map.

Definition at line 25 of file OriginDestinationSetTypes.hpp.

22.2.1.22 typedef std::list<ReachableUniverse*> AIRSCHED::ReachableUniverseList_T

Define the reachable-universe list.

Definition at line 18 of file ReachableUniverseTypes.hpp.

22.2.1.23 typedef std::map<const stdair::MapKey_T, ReachableUniverse*> AIRSCHED::ReachableUniverseMap_T

Define the reachable-universe map.

Definition at line 25 of file ReachableUniverseTypes.hpp.

22.2.1.24 typedef std::vector<SegmentCabinStruct> AIRSCHED::SegmentCabinStructList_T

List of SegmentCabin-Detail strucutres.

Definition at line 41 of file SegmentCabinStruct.hpp.

22.2.1.25 typedef std::list<SegmentPathPeriod*> AIRSCHED::SegmentPathPeriodList_T

Define the segment path list.

Definition at line 20 of file SegmentPathPeriodTypes.hpp.

22.2.1.26 typedef std::multimap<const stdair::MapKey_T, SegmentPathPeriod*> AIRSCHED::SegmentPathPeriod-Multimap_T

Define the segment path map.

Definition at line 27 of file SegmentPathPeriodTypes.hpp.

22.2.1.27 typedef std::vector<const SegmentPathPeriod*> AIRSCHED::SegmentPathPeriodLightList_T

Define the (temporary) containers of segment path period.

Definition at line 30 of file SegmentPathPeriodTypes.hpp.

22.2.1.28 typedef std::vector<SegmentPathPeriodLightList_T> AIRSCHED::SegmentPathPeriodListList_T

Definition at line 31 of file SegmentPathPeriodTypes.hpp.

22.2.1.29 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.

22.2.1.30 typedef std::vector<SegmentStruct> AIRSCHED::SegmentStructList_T

List of Segment strucutres.

Definition at line 44 of file SegmentStruct.hpp.

22.2.2 Function Documentation

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

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

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

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

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

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

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

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

22.2.2.9 template void **AIRSCHED::SegmentPathPeriod::serialize**< ba::text_oarchive > (ba::text_oarchive & , unsigned *int*)

22.2.2.10 template void **AIRSCHED::SegmentPathPeriod::serialize**< ba::text_iarchive > (ba::text_iarchive & , unsigned *int*)

22.2.2.11 template void **AIRSCHED::SegmentPathPeriodKey::serialize**< ba::text_oarchive > (ba::text_oarchive & , unsigned *int*)

22.2.2.12 template void **AIRSCHED::SegmentPathPeriodKey::serialize**< ba::text_iarchive > (ba::text_iarchive & , unsigned *int*)

22.2.3 Variable Documentation

22.2.3.1 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.

22.3 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

22.3.1 Function Documentation

22.3.1.1 chset_t AIRSCHED::OnDParserHelper::alpha_cap_set_p ("A-Z")

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

22.3.1.2 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().

22.3.1.3 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().

22.3.1.4 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().

22.3.1.5 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().

22.3.1.6 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().

22.3.1.7 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().

22.3.1.8 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().

22.3.1.9 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().

22.3.1.10 chset_t AIRSCHED::OnDParserHelper::class_code_p ("A-Z")

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

Referenced by AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >::definition().

22.3.2 Variable Documentation

22.3.2.1 uint2_p_t AIRSCHED::OnDParserHelper::uint2_p

2-digit-integer parser

Definition at line 215 of file OnDParserHelper.cpp.

22.3.2.2 uint4_p_t AIRSCHED::OnDParserHelper::uint4_p

4-digit-integer parser

Definition at line 218 of file OnDParserHelper.cpp.

22.3.2.3 uint1_4_p_t AIRSCHED::OnDParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 221 of file OnDParserHelper.cpp.

22.4 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

22.4.1 Function Documentation

22.4.1.1 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().

22.4.1.2 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().

22.4.1.3 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().

22.4.1.4 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().

22.4.1.5 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().

22.4.1.6 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().

22.4.1.7 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().

22.4.1.8 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().

22.4.1.9 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().

22.4.1.10 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().

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

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

Referenced by AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

22.4.1.12 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().

22.4.2 Variable Documentation

22.4.2.1 int1_p_t AIRSCHED::ScheduleParserHelper::int1_p

1-digit-integer parser

Definition at line 408 of file ScheduleParserHelper.cpp.

Referenced by AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

22.4.2.2 uint2_p_t AIRSCHED::ScheduleParserHelper::uint2_p

2-digit-integer parser

Definition at line 411 of file ScheduleParserHelper.cpp.

22.4.2.3 uint4_p_t AIRSCHED::ScheduleParserHelper::uint4_p

4-digit-integer parser

Definition at line 414 of file ScheduleParserHelper.cpp.

22.4.2.4 uint1_4_p_t AIRSCHED::ScheduleParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 417 of file ScheduleParserHelper.cpp.

22.4.2.5 int1_p_t AIRSCHED::ScheduleParserHelper::family_code_p

Family code parser

Definition at line 453 of file ScheduleParserHelper.cpp.

Referenced by AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

22.5 boost Namespace Reference

Forward declarations.

Namespaces

- namespace serialization

22.5.1 Detailed Description

Forward declarations.

22.6 boost::serialization Namespace Reference

22.7 stdair Namespace Reference

Forward declarations.

22.7.1 Detailed Description

Forward declarations.

23 Class Documentation

23.1 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

23.1.1 Detailed Description

Airline.

Definition at line 52 of file BookingRequestParser.hpp.

23.1.2 Constructor & Destructor Documentation

23.1.2.1 airsched::Airline_T::Airline_T ( ) [inline]

Constructor.

Definition at line 58 of file BookingRequestParser.hpp.

23.1.3 Member Function Documentation

23.1.3.1 void airsched::Airline_T::display ( ) const [inline]

Definition at line 60 of file BookingRequestParser.hpp.

References _code, _isPreferred, and _name.

Referenced by airsched::SearchString_T::display().

23.1.4 Member Data Documentation

23.1.4.1 bool airsched::Airline_T::_isPreferred

Definition at line 54 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_airline_sign::operator()().

23.1.4.2 std::string airsched::Airline_T::_name

Definition at line 55 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_airline_name::operator()().

23.1.4.3 std::string airsched::Airline_T::_code

Definition at line 56 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_airline_code::operator()().

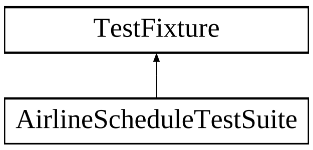
The documentation for this struct was generated from the following file:

- airsched/batches/BookingRequestParser.hpp

23.2 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

23.2.1 Detailed Description

Definition at line 6 of file AirlineScheduleTestSuite.hpp.

23.2.2 Constructor & Destructor Documentation

23.2.2.1 AirlineScheduleTestSuite::AirlineScheduleTestSuite ( )

Constructor.

23.2.3 Member Function Documentation

23.2.3.1 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.

23.2.3.2 void AirlineScheduleTestSuite::scheduleParsing ( )

23.2.4 Member Data Documentation

23.2.4.1 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

23.3 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

23.3.1 Detailed Description

Interface for the AirSched Services.

Definition at line 32 of file AIRSCHED_Service.hpp.

23.3.2 Constructor & Destructor Documentation

23.3.2.1 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

const	stdair::BasLogParams& Parameters for the output log stream.
const	stdair::BasDBParams& Parameters for the database access.

Definition at line 62 of file AIRSCHED_Service.cpp.

23.3.2.2 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

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

Definition at line 42 of file AIRSCHED_Service.cpp.

23.3.2.3 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

stdair::STDAIR_-ServicePtr_T	Reference on the STDAIR service.

Definition at line 84 of file AIRSCHED_Service.cpp.

23.3.2.4 AIRSCHED::AIRSCHED_Service::∼AIRSCHED_Service ( )

Destructor.

Definition at line 100 of file AIRSCHED_Service.cpp.

23.3.3 Member Function Documentation

23.3.3.1 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

const	stdair::Filename_T& Filename of the input schedule file.

Definition at line 178 of file AIRSCHED_Service.cpp.

References AIRSCHED::ScheduleParser::generateInventories().

Referenced by main(), and parseAndLoad().

23.3.3.2 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

const	stdair::Filename_T& Filename of the input schedule file.
const	stdair::Filename_T& Filename of the input O&D file.

Definition at line 199 of file AIRSCHED_Service.cpp.

References AIRSCHED::OnDParser::generateOnDPeriods(), and parseAndLoad().

23.3.3.3 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().

23.3.3.4 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().

23.3.3.5 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.

23.3.3.6 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

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

Returns

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

Definition at line 274 of file AIRSCHED_Service.cpp.

23.3.3.7 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.

23.3.3.8 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

const	stdair::AirlineCode_T& Airline code of the flight period to display.
const	stdair::FlightNumber_T& Flight number of the flight to display.
const	stdair::Date_T& Departure date of a flight within the flight-period to display.

Returns

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

Definition at line 318 of file AIRSCHED_Service.cpp.

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

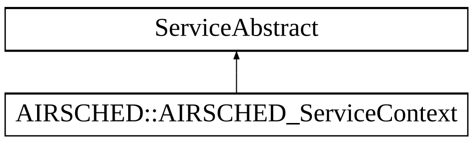
- airsched/AIRSCHED_Service.hpp
- airsched/service/AIRSCHED_Service.cpp

23.4 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

23.4.1 Detailed Description

Class holding the context of the AirSched services.

Definition at line 22 of file AIRSCHED_ServiceContext.hpp.

23.4.2 Friends And Related Function Documentation

23.4.2.1 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.

23.4.2.2 friend class **FacAIRSCHEDServiceContext** `[friend]`

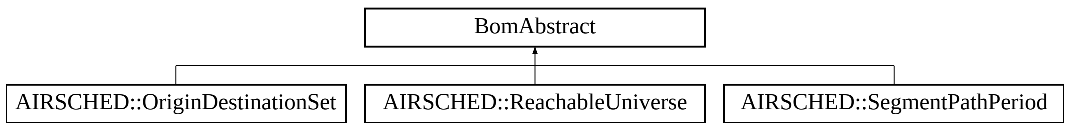
Definition at line 29 of file AIRSCHED_ServiceContext.hpp.

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

- airsched/service/AIRSCHED_ServiceContext.hpp
- airsched/service/AIRSCHED_ServiceContext.cpp

23.5 BomAbstract Class Reference

Inheritance diagram for BomAbstract:


BomAbstract
AIRSCHED::OriginDestinationSet
AIRSCHED::ReachableUniverse
AIRSCHED::SegmentPathPeriod

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

- airsched/bom/OriginDestinationSet.hpp

23.6 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 &)

23.6.1 Detailed Description

Utility class to display AirSched objects with a pretty format.

Definition at line 26 of file BomDisplay.hpp.

23.6.2 Member Function Documentation

23.6.2.1 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

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

Definition at line 43 of file BomDisplay.cpp.

23.6.2.2 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

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

Definition at line 81 of file BomDisplay.cpp.

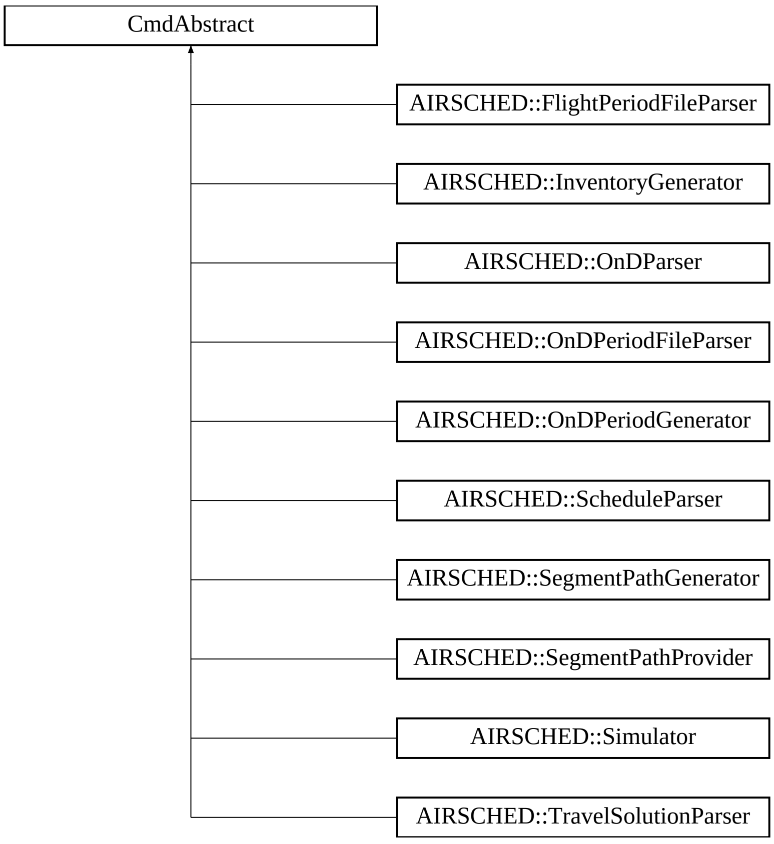
References AIRSCHED::ReachableUniverse::toString().

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

- airsched/bom/BomDisplay.hpp
- airsched/bom/BomDisplay.cpp

23.7 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:


CmdAbstract
AIRSCHED::FlightPeriodFileParser
AIRSCHED::InventoryGenerator
AIRSCHED::OnDParser
AIRSCHED::OnDPeriodFileParser
AIRSCHED::OnDPeriodGenerator
AIRSCHED::ScheduleParser
AIRSCHED::SegmentPathGenerator
AIRSCHED::SegmentPathProvider
AIRSCHED::Simulator
AIRSCHED::TravelSolutionParser

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

- airsched/command/OnDPeriodGenerator.hpp

23.8 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

23.8.1 Detailed Description

Date.

Definition at line 27 of file BookingRequestParser.hpp.

23.8.2 Constructor & Destructor Documentation

23.8.2.1 airsched::Date_T::Date_T ( ) [inline]

Constructor.

Definition at line 35 of file BookingRequestParser.hpp.

23.8.3 Member Function Documentation

23.8.3.1 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().

23.8.3.2 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()().

23.8.4 Member Data Documentation

23.8.4.1 boost::gregorian::date airsched::Date_T::_date

Definition at line 29 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_date::operator()().

23.8.4.2 unsigned int airsched::Date_T::_reldays

Definition at line 30 of file BookingRequestParser.hpp.

Referenced by display().

23.8.4.3 unsigned int airsched::Date_T::_day

Definition at line 31 of file BookingRequestParser.hpp.

Referenced by display(), and getDate().

23.8.4.4 unsigned int airsched::Date_T::_month

Definition at line 32 of file BookingRequestParser.hpp.

Referenced by display(), and getDate().

23.8.4.5 unsigned int airsched::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:

- airsched/batches/BookingRequestParser.hpp

23.9 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

23.9.1 Detailed Description

template<typename ScannerT>struct airsched::SearchStringParser::definition< ScannerT >

Definition at line 259 of file BookingRequestParser.cpp.

23.9.2 Constructor & Destructor Documentation

23.9.2.1 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.

23.9.3 Member Function Documentation

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

Definition at line 366 of file BookingRequestParser.cpp.

23.9.4 Member Data Documentation

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

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

Definition at line 360 of file BookingRequestParser.cpp.

23.9.4.17 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

23.10 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

23.10.1 Detailed Description

template<typename ScannerT>struct AIRSCHED::OnDParserHelper::OnDParser::definition< ScannerT >

Definition at line 133 of file OnDParserHelper.hpp.

23.10.2 Constructor & Destructor Documentation

23.10.2.1 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().

23.10.3 Member Function Documentation

23.10.3.1 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.

23.10.4 Member Data Documentation

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

Definition at line 137 of file OnDParserHelper.hpp.

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

Definition at line 137 of file OnDParserHelper.hpp.

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

Definition at line 137 of file OnDParserHelper.hpp.

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

Definition at line 137 of file OnDParserHelper.hpp.

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

Definition at line 137 of file OnDParserHelper.hpp.

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

Definition at line 137 of file OnDParserHelper.hpp.

23.10.4.7 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

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

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

Public Member Functions

- definition (FlightPeriodParser const &self)
- boost::spirit::classic::rule
  < ScannerT > const & start () const

Public Attributes

- boost::spirit::classic::rule
  < ScannerT > flight_period_list
- boost::spirit::classic::rule
  < ScannerT > flight_period
- boost::spirit::classic::rule
  < ScannerT > not_to_be_parsed
- boost::spirit::classic::rule
  < ScannerT > flight_period_end
- boost::spirit::classic::rule
  < ScannerT > flight_key
- boost::spirit::classic::rule
  < ScannerT > airline_code
- boost::spirit::classic::rule
  < ScannerT > flight_number

- boost::spirit::classic::rule
  < ScannerT > date
- boost::spirit::classic::rule
  < ScannerT > dow
- boost::spirit::classic::rule
  < ScannerT > time
- boost::spirit::classic::rule
  < ScannerT > date_offset
- boost::spirit::classic::rule
  < ScannerT > leg
- boost::spirit::classic::rule
  < ScannerT > leg_key
- boost::spirit::classic::rule
  < ScannerT > 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

23.11.1 Detailed Description

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

Definition at line 255 of file ScheduleParserHelper.hpp.

23.11.2 Constructor & Destructor Documentation

23.11.2.1 template<typename ScannerT > AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition (FlightPeriodParser const & *self*)

Definition at line 474 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().

23.11.3 Member Function Documentation

23.11.3.1 template< typename ScannerT > bsc::rule< ScannerT > const & AIRSCHED::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::start ( ) const

Entry point of the parser.

Definition at line 621 of file ScheduleParserHelper.cpp.

23.11.4 Member Data Documentation

23.11.4.1 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::flight_period_list

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.2 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::flight_period

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.3 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::not_to_be_parsed

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.4 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::flight_period_end

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.5 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::flight_key

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.6 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::airline_code

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.7 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::flight_number

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.8 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::date

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.9 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParserHelper-::FlightPeriodParser::definition< ScannerT >::dow

Definition at line 259 of file ScheduleParserHelper.hpp.

23.11.4.10 template< typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRSCHED::ScheduleParser-Helper::FlightPeriodParser::definition< ScannerT >::time

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

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

Definition at line 259 of file ScheduleParserHelper.hpp.

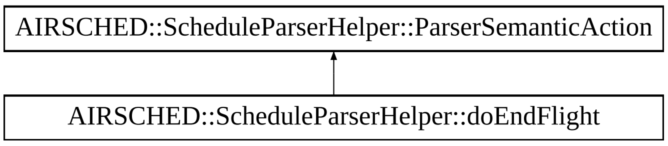
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.12 AIRSCHED::ScheduleParserHelper::doEndFlight Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::doEndFlight:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
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

23.12.1 Detailed Description

Mark the end of the flight-period parsing.

Definition at line 192 of file ScheduleParserHelper.hpp.

23.12.2 Constructor & Destructor Documentation

23.12.2.1 AIRSCHED::ScheduleParserHelper::doEndFlight::doEndFlight (stdair::BomRoot & *ioBomRoot,* FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 375 of file ScheduleParserHelper.cpp.

23.12.3 Member Function Documentation

23.12.3.1 void AIRSCHED::ScheduleParserHelper::doEndFlight::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 383 of file ScheduleParserHelper.cpp.

References _bomRoot, AIRSCHED::LegStruct::_cabinList, AIRSCHED::ScheduleParserHelper::ParserSemantic-Action::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itLeg, AIRSCHED::FlightPeriodStruct::_legAlreadyDefined, AIRSCHED::FlightPeriodStruct::_legList, and AIRSCHED::FlightPeriodStruct::describe().

23.12.4 Member Data Documentation

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

Actor Specific Context.

Definition at line 198 of file ScheduleParserHelper.hpp.

Referenced by operator()().

23.12.4.2 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper-::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED-::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime-::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper-::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRS-CHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegment-Specificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRS-CHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::store-SegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHE-D::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses-::operator()(), and operator()().

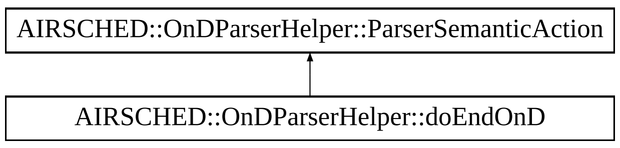
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.13 AIRSCHED::OnDParserHelper::doEndOnD Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::doEndOnD:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::doEndOnD

Public Member Functions

- doEndOnD (stdair::BomRoot &, OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & _bomRoot

- OnDPeriodStruct & _onDPeriod

23.13.1 Detailed Description

Mark the end of the O&D parsing.

Definition at line 106 of file OnDParserHelper.hpp.

23.13.2 Constructor & Destructor Documentation

23.13.2.1 AIRSCHED::OnDParserHelper::doEndOnD::doEndOnD (stdair::BomRoot & *ioBomRoot,* OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 193 of file OnDParserHelper.cpp.

23.13.3 Member Function Documentation

23.13.3.1 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.

23.13.4 Member Data Documentation

23.13.4.1 stdair::BomRoot& AIRSCHED::OnDParserHelper::doEndOnD::_bomRoot

Actor Specific Context.

Definition at line 112 of file OnDParserHelper.hpp.

Referenced by operator()().

23.13.4.2 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

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

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

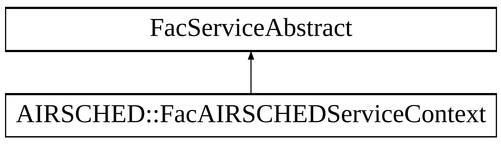
- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.14 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 ()

23.14.1 Detailed Description

Factory for the service context.

Definition at line 19 of file FacAIRSCHEDServiceContext.hpp.

23.14.2 Constructor & Destructor Documentation

23.14.2.1 AIRSCHED::FacAIRSCHEDServiceContext::∼FacAIRSCHEDServiceContext ( )

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacAIRSCHEDServiceContext::instance().

Definition at line 17 of file FacAIRSCHEDServiceContext.cpp.

23.14.2.2 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().

23.14.3 Member Function Documentation

23.14.3.1 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().

23.14.3.2 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.

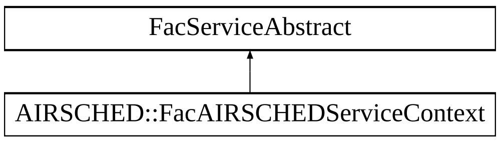
Definition at line 34 of file FacAIRSCHEDServiceContext.cpp.

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

- airsched/factory/FacAIRSCHEDServiceContext.hpp
- airsched/factory/FacAIRSCHEDServiceContext.cpp

23.15 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



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

- airsched/factory/FacAIRSCHEDServiceContext.hpp

23.16 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

23.16.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file FacServiceAbstract.hpp.

23.16.2 Member Typedef Documentation

23.16.2.1 typedef std::vector<ServiceAbstract*> AIRSCHED::FacServiceAbstract::ServicePool_T

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

23.16.3 Constructor & Destructor Documentation

23.16.3.1 AIRSCHED::FacServiceAbstract::∼FacServiceAbstract ( ) `[virtual]`

Destructor.

Definition at line 13 of file FacServiceAbstract.cpp.

References clean().

23.16.3.2 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.

23.16.4 Member Function Documentation

23.16.4.1 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().

23.16.5 Member Data Documentation

23.16.5.1 ServicePool_T AIRSCHED::FacServiceAbstract::_pool `[protected]`

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by clean().

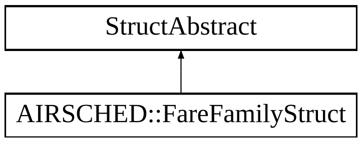
The documentation for this class was generated from the following files:

- airsched/factory/FacServiceAbstract.hpp
- airsched/factory/FacServiceAbstract.cpp

23.17 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

23.17.1 Detailed Description

Utility Structure for the parsing of fare family details.

Definition at line 17 of file FareFamilyStruct.hpp.

23.17.2 Constructor & Destructor Documentation

23.17.2.1 AIRSCHED::FareFamilyStruct::FareFamilyStruct (const stdair::FamilyCode_T & *iFamilyCode,* const stdair::ClassList_String_T & *iClasses*)

Constructors.

Definition at line 14 of file FareFamilyStruct.cpp.

23.17.3 Member Function Documentation

23.17.3.1 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.

23.17.4 Member Data Documentation

23.17.4.1 stdair::FamilyCode_T AIRSCHED::FareFamilyStruct::_familyCode

Definition at line 19 of file FareFamilyStruct.hpp.

Referenced by describe().

23.17.4.2 stdair::ClassList_String_T AIRSCHED::FareFamilyStruct::_classes

Definition at line 20 of file FareFamilyStruct.hpp.

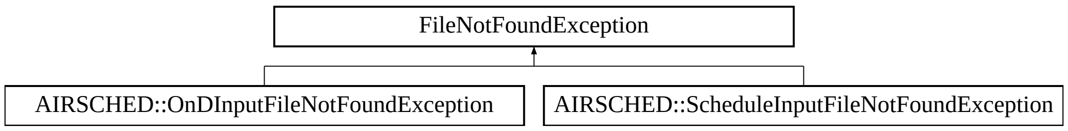
Referenced by describe().

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

- airsched/bom/FareFamilyStruct.hpp
- airsched/bom/FareFamilyStruct.cpp

23.18 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException:


FileNotFoundException
AIRSCHED::OnDInputFileNotFoundException
AIRSCHED::ScheduleInputFileNotFoundException

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

- airsched/AIRSCHED_Types.hpp

23.19 AIRSCHED::FlagSaver Struct Reference

Public Member Functions

- FlagSaver (std::ostream &oStream)
- ∼FlagSaver ()

23.19.1 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.

23.19.2 Constructor & Destructor Documentation

23.19.2.1 AIRSCHED::FlagSaver::FlagSaver (std::ostream & *oStream*) `[inline]`

Constructor.

Definition at line 25 of file BomDisplay.cpp.

23.19.2.2 AIRSCHED::FlagSaver::∼FlagSaver ( ) `[inline]`

Destructor.

Definition at line 30 of file BomDisplay.cpp.

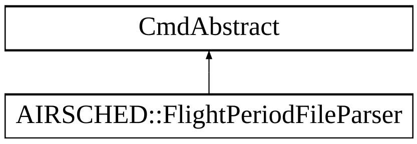
The documentation for this struct was generated from the following file:

- airsched/bom/BomDisplay.cpp

23.20 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 ()

23.20.1 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 291 of file ScheduleParserHelper.hpp.

23.20.2 Constructor & Destructor Documentation

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

Constructor.

Definition at line 636 of file ScheduleParserHelper.cpp.

23.20.3 Member Function Documentation

23.20.3.1 bool AIRSCHED::FlightPeriodFileParser::generateInventories ( )

Parse the input file and generate the Inventories.

Definition at line 673 of file ScheduleParserHelper.cpp.

Referenced by AIRSCHED::ScheduleParser::generateInventories().

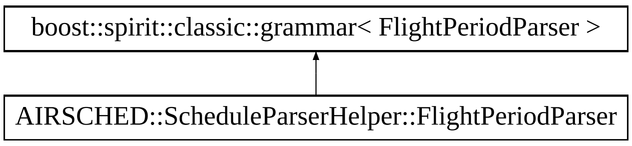
The documentation for this class was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.21 AIRSCHED::ScheduleParserHelper::FlightPeriodParser Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::FlightPeriodParser:


boost::spirit::classic::grammar< FlightPeriodParser >
AIRSCHED::ScheduleParserHelper::FlightPeriodParser

Classes

- struct definition

Public Member Functions

- FlightPeriodParser (stdair::BomRoot &, FlightPeriodStruct &)

Public Attributes

- stdair::BomRoot & _bomRoot
- FlightPeriodStruct & _flightPeriod

23.21.1 Detailed Description

AirlineCode; FlightNumber; DateRangeStart; DateRangeEnd; DOW; (list) BoardingPoint; OffPoint; BoardingTime; DateOffset; OffTime; ElapsedTime; (list) CabinCode; Capacity; SegmentSpecificty (0 or 1); (list) (optional Boarding-Point; 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; YHBKLMNOP-QRSTVWX; 3; YHBKLMNOPQRSTVWX BA; 9; 2007-04-20; 2007-04-30; 1111100; LHR; SIN; 22:00; +1; 15:15; 11:15; C; 15; M; 310; SIN; SYD; 18:10; +1; 06:05; 08:55; C; 25; M; 260; 1; LHR; SIN; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTVWX SIN; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTVWX LHR; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTV-WX

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

Definition at line 249 of file ScheduleParserHelper.hpp.

23.21.2 Constructor & Destructor Documentation

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

Definition at line 465 of file ScheduleParserHelper.cpp.

23.21.3 Member Data Documentation

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

Definition at line 272 of file ScheduleParserHelper.hpp.

23.21.3.2 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::FlightPeriodParser::_flightPeriod

Definition at line 273 of file ScheduleParserHelper.hpp.

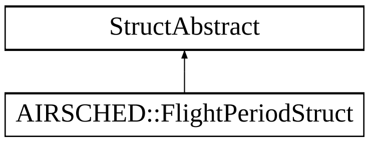
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.22 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

23.22.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

Definition at line 26 of file FlightPeriodStruct.hpp.

23.22.2 Constructor & Destructor Documentation

23.22.2.1 AIRSCHED::FlightPeriodStruct::FlightPeriodStruct ( )

Constructor.

Definition at line 17 of file FlightPeriodStruct.cpp.

23.22.3 Member Function Documentation

23.22.3.1 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::storeDateRangeStart::operator()(), and AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()().

23.22.3.2 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::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), and AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()().

23.22.3.3 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()().

23.22.3.4 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::storeLegBoardingPoint::operator()(), and AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()().

23.22.3.5 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()().

23.22.3.6 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()().

23.22.3.7 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.

23.22.3.8 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 build-SegmentCabins() 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()().

23.22.3.9 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.

23.22.4 Member Data Documentation

23.22.4.1 stdair::AirlineCode_T AIRSCHED::FlightPeriodStruct::_airlineCode

Definition at line 84 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()().

23.22.4.2 stdair::FlightNumber_T AIRSCHED::FlightPeriodStruct::_flightNumber

Definition at line 85 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()().

23.22.4.3 stdair::DatePeriod_T AIRSCHED::FlightPeriodStruct::_dateRange

Definition at line 86 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()().

23.22.4.4 stdair::DoWStruct AIRSCHED::FlightPeriodStruct::_dow

Definition at line 87 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::ScheduleParserHelper::storeDow::operator()().

23.22.4.5 LegStructList_T AIRSCHED::FlightPeriodStruct::_legList

Definition at line 88 of file FlightPeriodStruct.hpp.

Referenced by describe(), AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::-ScheduleParserHelper::storeLegBoardingPoint::operator()(), and AIRSCHED::ScheduleParserHelper::doEnd-Flight::operator()().

23.22.4.6 SegmentStructList_T AIRSCHED::FlightPeriodStruct::_segmentList

Definition at line 89 of file FlightPeriodStruct.hpp.

Referenced by addFareFamily(), addSegmentCabin(), buildSegments(), and describe().

23.22.4.7 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::storeLegBoardingPoint::operator()(), and AIRSCHED::-ScheduleParserHelper::doEndFlight::operator()().

23.22.4.8 LegStruct AIRSCHED::FlightPeriodStruct::_itLeg

Definition at line 94 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::Schedule-ParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::store-ElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), and AIRSCHED::-ScheduleParserHelper::doEndFlight::operator()().

23.22.4.9 LegCabinStruct AIRSCHED::FlightPeriodStruct::_itLegCabin

Definition at line 95 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), and AIRSCHED::Schedule-ParserHelper::storeCapacity::operator()().

23.22.4.10 stdair::Date_T AIRSCHED::FlightPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 98 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), and AIRSCHED::Schedule-ParserHelper::storeDateRangeEnd::operator()().

23.22.4.11 stdair::Date_T AIRSCHED::FlightPeriodStruct::_dateRangeEnd

Definition at line 99 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()().

23.22.4.12 unsigned int AIRSCHED::FlightPeriodStruct::_itYear

Definition at line 100 of file FlightPeriodStruct.hpp.

Referenced by getDate().

23.22.4.13 unsigned int AIRSCHED::FlightPeriodStruct::_itMonth

Definition at line 101 of file FlightPeriodStruct.hpp.

Referenced by getDate().

23.22.4.14 unsigned int AIRSCHED::FlightPeriodStruct::_itDay

Definition at line 102 of file FlightPeriodStruct.hpp.

Referenced by getDate().

23.22.4.15 int AIRSCHED::FlightPeriodStruct::_dateOffset

Definition at line 103 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParser-Helper::storeOffTime::operator()(), and AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()().

23.22.4.16 long AIRSCHED::FlightPeriodStruct::_itHours

Staging Time.

Definition at line 106 of file FlightPeriodStruct.hpp.

Referenced by getTime().

23.22.4.17 long AIRSCHED::FlightPeriodStruct::_itMinutes

Definition at line 107 of file FlightPeriodStruct.hpp.

Referenced by getTime().

23.22.4.18 long AIRSCHED::FlightPeriodStruct::_itSeconds

Definition at line 108 of file FlightPeriodStruct.hpp.

Referenced by getTime(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED-::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeBoarding-Time::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), and AIRSCHED::Schedule-ParserHelper::storeElapsedTime::operator()().

23.22.4.19 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()().

23.22.4.20 AirportOrderedList_T AIRSCHED::FlightPeriodStruct::_airportOrderedList

Definition at line 113 of file FlightPeriodStruct.hpp.

Referenced by addAirport(), buildSegments(), and AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity-::operator()().

23.22.4.21 bool AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 116 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::Schedule-ParserHelper::storeClasses::operator()(), and AIRSCHED::ScheduleParserHelper::storeFClasses::operator()().

23.22.4.22 SegmentStruct AIRSCHED::FlightPeriodStruct::_itSegment

Definition at line 117 of file FlightPeriodStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::-ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses-::operator()(), and AIRSCHED::ScheduleParserHelper::storeFClasses::operator()().

23.22.4.23 SegmentCabinStruct AIRSCHED::FlightPeriodStruct::_itSegmentCabin

Definition at line 118 of file FlightPeriodStruct.hpp.

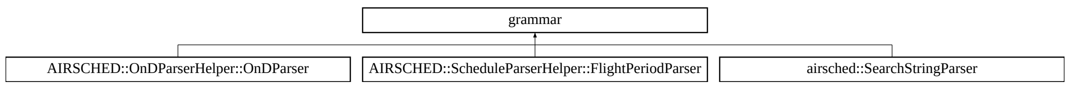
Referenced by AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::Schedule-ParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), and AIRSCHED::ScheduleParserHelper::storeFClasses::operator()().

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

- airsched/bom/FlightPeriodStruct.hpp
- airsched/bom/FlightPeriodStruct.cpp

23.23 grammar Class Reference

Inheritance diagram for grammar:


grammar
AIRSCHED::OnDParserHelper::OnDParser
AIRSCHED::ScheduleParserHelper::FlightPeriodParser
airsched::SearchStringParser

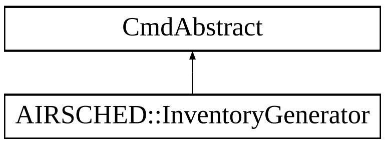
The documentation for this class was generated from the following file:

- airsched/command/OnDParserHelper.hpp

23.24 AIRSCHED::InventoryGenerator Class Reference

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

Inheritance diagram for AIRSCHED::InventoryGenerator:



Friends

- class FlightPeriodFileParser
- class FFFlightPeriodFileParser
- struct ScheduleParserHelper::doEndFlight
- class ScheduleParser

23.24.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

Definition at line 31 of file InventoryGenerator.hpp.

23.24.2 Friends And Related Function Documentation

23.24.2.1 friend class **FlightPeriodFileParser** `[friend]`

Definition at line 35 of file InventoryGenerator.hpp.

23.24.2.2 friend class FFFlightPeriodFileParser `[friend]`

Definition at line 36 of file InventoryGenerator.hpp.

23.24.2.3 friend struct **ScheduleParserHelper::doEndFlight** `[friend]`

Definition at line 37 of file InventoryGenerator.hpp.

23.24.2.4 friend class **ScheduleParser** `[friend]`

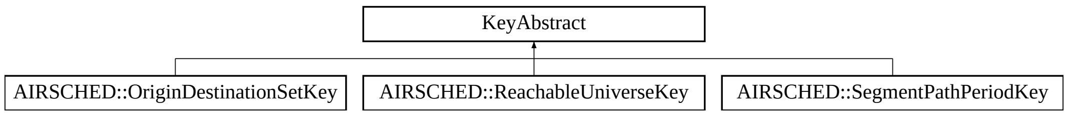
Definition at line 38 of file InventoryGenerator.hpp.

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

- airsched/command/InventoryGenerator.hpp
- airsched/command/InventoryGenerator.cpp

23.25 KeyAbstract Class Reference

Inheritance diagram for KeyAbstract:


KeyAbstract
AIRSCHED::OriginDestinationSetKey
AIRSCHED::ReachableUniverseKey
AIRSCHED::SegmentPathPeriodKey

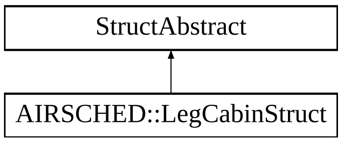
The documentation for this class was generated from the following file:

- airsched/bom/ReachableUniverseKey.hpp

23.26 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

23.26.1 Detailed Description

Utility Structure for the parsing of LegCabin details.

Definition at line 22 of file LegCabinStruct.hpp.

23.26.2 Member Function Documentation

23.26.2.1 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.

23.26.2.2 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().

23.26.3 Member Data Documentation

23.26.3.1 stdair::CabinCode_T AIRSCHED::LegCabinStruct::_cabinCode

Definition at line 24 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()().

23.26.3.2 stdair::CabinCapacity_T AIRSCHED::LegCabinStruct::_capacity

Definition at line 25 of file LegCabinStruct.hpp.

Referenced by describe(), fill(), and AIRSCHED::ScheduleParserHelper::storeCapacity::operator()().

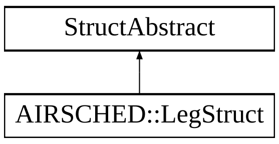
The documentation for this struct was generated from the following files:

- airsched/bom/LegCabinStruct.hpp
- airsched/bom/LegCabinStruct.cpp

23.27 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

23.27.1 Detailed Description

Utility Structure for the parsing of Leg structures.

Definition at line 24 of file LegStruct.hpp.

23.27.2 Constructor & Destructor Documentation

23.27.2.1 AIRSCHED::LegStruct::LegStruct ( )

Default Constructor.

Definition at line 16 of file LegStruct.cpp.

23.27.3 Member Function Documentation

23.27.3.1 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.

23.27.3.2 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().

23.27.4 Member Data Documentation

23.27.4.1 stdair::AirportCode_T AIRSCHED::LegStruct::_boardingPoint

Definition at line 26 of file LegStruct.hpp.

Referenced by describe(), AIRSCHED::SegmentPeriodHelper::fill(), and AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()().

23.27.4.2 stdair::DateOffset_T AIRSCHED::LegStruct::_boardingDateOffset

Definition at line 27 of file LegStruct.hpp.

Referenced by describe(), fill(), and AIRSCHED::ScheduleParserHelper::storeOffTime::operator()().

23.27.4.3 stdair::Duration_T AIRSCHED::LegStruct::_boardingTime

Definition at line 28 of file LegStruct.hpp.

Referenced by describe(), fill(), and AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()().

23.27.4.4 stdair::AirportCode_T AIRSCHED::LegStruct::_offPoint

Definition at line 29 of file LegStruct.hpp.

Referenced by describe(), AIRSCHED::SegmentPeriodHelper::fill(), fill(), and AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()().

23.27.4.5 stdair::DateOffset_T AIRSCHED::LegStruct::_offDateOffset

Definition at line 30 of file LegStruct.hpp.

Referenced by describe(), AIRSCHED::SegmentPeriodHelper::fill(), fill(), and AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()().

23.27.4.6 stdair::Duration_T AIRSCHED::LegStruct::_offTime

Definition at line 31 of file LegStruct.hpp.

Referenced by describe(), AIRSCHED::SegmentPeriodHelper::fill(), fill(), and AIRSCHED::ScheduleParserHelper::storeOffTime::operator()().

23.27.4.7 stdair::Duration_T AIRSCHED::LegStruct::_elapsed

Definition at line 32 of file LegStruct.hpp.

Referenced by describe(), fill(), and AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()().

23.27.4.8 LegCabinStructList_T AIRSCHED::LegStruct::_cabinList

Definition at line 33 of file LegStruct.hpp.

Referenced by describe(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

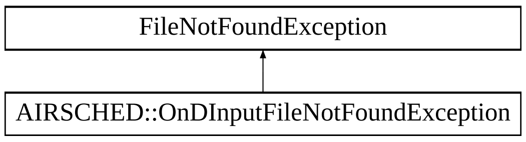
The documentation for this struct was generated from the following files:

- airsched/bom/LegStruct.hpp
- airsched/bom/LegStruct.cpp

23.28 AIRSCHED::OnDInputFileNotFoundException Class Reference

`#include <airsched/AIRSCHED_Types.hpp>`

Inheritance diagram for AIRSCHED::OnDInputFileNotFoundException:



Public Member Functions

- OnDInputFileNotFoundException (const std::string &iWhat)

23.28.1 Detailed Description

The O&D input file cannot be retrieved.

Definition at line 35 of file AIRSCHED_Types.hpp.

23.28.2 Constructor & Destructor Documentation

23.28.2.1 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:

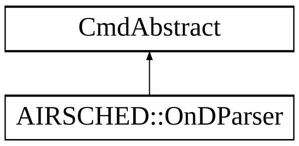
- airsched/AIRSCHED_Types.hpp

23.29 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 &)

23.29.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 23 of file OnDParser.hpp.

23.29.2 Member Function Documentation

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

Parse the CSV file describing the O&D.

Parameters

const	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().

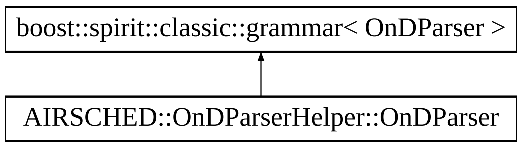
The documentation for this class was generated from the following files:

- airsched/command/OnDParser.hpp
- airsched/command/OnDParser.cpp

23.30 AIRSCHED::OnDParserHelper::OnDParser Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::OnDParser:


boost::spirit::classic::grammar< OnDParser >
AIRSCHED::OnDParserHelper::OnDParser

Classes

- struct definition

Public Member Functions

- OnDParser (stdair::BomRoot &, OnDPeriodStruct &)

Public Attributes

- stdair::BomRoot & _bomRoot
- OnDPeriodStruct & _onDPeriod

23.30.1 Detailed Description

Fares: AirlineCode; OriginCity; DestinationCity; DepartureDate-Range(FirstDate; LastDate); Airline; Class; BA; N-CE; 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.

23.30.2 Constructor & Destructor Documentation

23.30.2.1 AIRSCHED::OnDParserHelper::OnDParser::OnDParser (stdair::BomRoot & *ioBomRoot,* OnDPeriodStruct & *ioOnDPeriod*)

Definition at line 261 of file OnDParserHelper.cpp.

23.30.3 Member Data Documentation

23.30.3.1 stdair::BomRoot& AIRSCHED::OnDParserHelper::OnDParser::_bomRoot

Definition at line 145 of file OnDParserHelper.hpp.

23.30.3.2 OnDPeriodStruct& AIRSCHED::OnDParserHelper::OnDParser::_onDPeriod

Definition at line 146 of file OnDParserHelper.hpp.

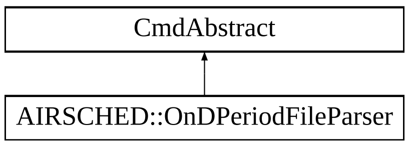
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.31 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 ()

23.31.1 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.

23.31.2 Constructor & Destructor Documentation

23.31.2.1 AIRSCHED::OnDPeriodFileParser::OnDPeriodFileParser (const stdair::Filename_T & *iFilename,* stdair::BomRoot & *ioBomRoot*)

Constructor.

Definition at line 342 of file OnDParserHelper.cpp.

23.31.3 Member Function Documentation

23.31.3.1 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:

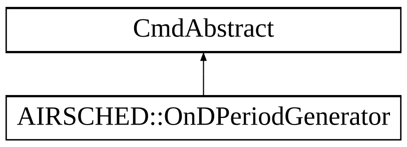
- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.32 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

23.32.1 Detailed Description

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

Definition at line 29 of file OnDPeriodGenerator.hpp.

23.32.2 Friends And Related Function Documentation

23.32.2.1 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.

23.32.2.2 friend struct **OnDParserHelper::doEndOnD** `[friend]`

Definition at line 36 of file OnDPeriodGenerator.hpp.

23.32.2.3 friend class **OnDParser** `[friend]`

Definition at line 37 of file OnDPeriodGenerator.hpp.

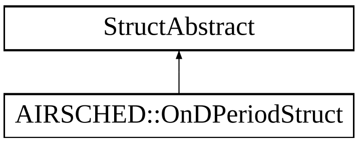
The documentation for this class was generated from the following files:

- airsched/command/OnDPeriodGenerator.hpp
- airsched/command/OnDPeriodGenerator.cpp

23.33 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

23.33.1 Detailed Description

Utility Structure for the parsing of FareRule structures.

Definition at line 15 of file OnDPeriodStruct.hpp.

23.33.2 Constructor & Destructor Documentation

23.33.2.1 AIRSCHED::OnDPeriodStruct::OnDPeriodStruct ( )

Default constructor.

Definition at line 17 of file OnDPeriodStruct.cpp.

23.33.3 Member Function Documentation

23.33.3.1 const stdair::AirlineCode_T & AIRSCHED::OnDPeriodStruct::getFirstAirlineCode ( ) const

Get the first airline code.

Definition at line 64 of file OnDPeriodStruct.cpp.

References _airlineCodeList.

23.33.3.2 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::storeDateRangeStart::operator()(), and AIRSCHED::OnDParser-Helper::storeDateRangeEnd::operator()().

23.33.3.3 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::storeStartRangeTime::operator()(), and AIRSCHED::OnDParser-Helper::storeEndRangeTime::operator()().

23.33.3.4 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.

23.33.3.5 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.

23.33.4 Member Data Documentation

23.33.4.1 stdair::AirportCode_T AIRSCHED::OnDPeriodStruct::_origin

Definition at line 41 of file OnDPeriodStruct.hpp.

Referenced by describe(), describeTSKey(), and AIRSCHED::OnDParserHelper::storeOrigin::operator()().

23.33.4.2 stdair::AirportCode_T AIRSCHED::OnDPeriodStruct::_destination

Definition at line 42 of file OnDPeriodStruct.hpp.

Referenced by describe(), describeTSKey(), and AIRSCHED::OnDParserHelper::storeDestination::operator()().

23.33.4.3 stdair::DatePeriod_T AIRSCHED::OnDPeriodStruct::_datePeriod

Definition at line 43 of file OnDPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()().

23.33.4.4 stdair::Duration_T AIRSCHED::OnDPeriodStruct::_timeRangeStart

Definition at line 44 of file OnDPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()().

23.33.4.5 stdair::Duration_T AIRSCHED::OnDPeriodStruct::_timeRangeEnd

Definition at line 45 of file OnDPeriodStruct.hpp.

Referenced by describe(), and AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()().

23.33.4.6 stdair::NbOfAirlines_T AIRSCHED::OnDPeriodStruct::_nbOfAirlines

Definition at line 46 of file OnDPeriodStruct.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), and AIRSCHED::OnDParserHelper::store-AirlineCode::operator()().

23.33.4.7 stdair::AirlineCode_T AIRSCHED::OnDPeriodStruct::_airlineCode

Definition at line 47 of file OnDPeriodStruct.hpp.

Referenced by describe(), describeTSKey(), AIRSCHED::OnDParserHelper::storeOrigin::operator()(), and AIRSC-HED::OnDParserHelper::storeAirlineCode::operator()().

23.33.4.8 stdair::ClassCode_T AIRSCHED::OnDPeriodStruct::_classCode

Definition at line 48 of file OnDPeriodStruct.hpp.

Referenced by describe(), describeTSKey(), AIRSCHED::OnDParserHelper::storeOrigin::operator()(), and AIRSC-HED::OnDParserHelper::storeClassCode::operator()().

23.33.4.9 stdair::AirlineCodeList_T AIRSCHED::OnDPeriodStruct::_airlineCodeList

Definition at line 49 of file OnDPeriodStruct.hpp.

Referenced by getFirstAirlineCode(), AIRSCHED::OnDParserHelper::storeOrigin::operator()(), and AIRSCHED::-

OnDParserHelper::storeAirlineCode::operator()().

23.33.4.10 stdair::ClassCodeList_T AIRSCHED::OnDPeriodStruct::_classCodeList

Definition at line 50 of file OnDPeriodStruct.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), and AIRSCHED::OnDParserHelper::store-ClassCode::operator()().

23.33.4.11 stdair::Date_T AIRSCHED::OnDPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 53 of file OnDPeriodStruct.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), and AIRSCHED::OnDParser-Helper::storeDateRangeEnd::operator()().

23.33.4.12 stdair::Date_T AIRSCHED::OnDPeriodStruct::_dateRangeEnd

Definition at line 54 of file OnDPeriodStruct.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()().

23.33.4.13 unsigned int AIRSCHED::OnDPeriodStruct::_itYear

Definition at line 55 of file OnDPeriodStruct.hpp.

Referenced by getDate().

23.33.4.14 unsigned int AIRSCHED::OnDPeriodStruct::_itMonth

Definition at line 56 of file OnDPeriodStruct.hpp.

Referenced by getDate().

23.33.4.15 unsigned int AIRSCHED::OnDPeriodStruct::_itDay

Definition at line 57 of file OnDPeriodStruct.hpp.

Referenced by getDate().

23.33.4.16 long AIRSCHED::OnDPeriodStruct::_itHours

Staging Time.

Definition at line 60 of file OnDPeriodStruct.hpp.

Referenced by getTime().

23.33.4.17 long AIRSCHED::OnDPeriodStruct::_itMinutes

Definition at line 61 of file OnDPeriodStruct.hpp.

Referenced by getTime().

23.33.4.18 long AIRSCHED::OnDPeriodStruct::_itSeconds

Definition at line 62 of file OnDPeriodStruct.hpp.

Referenced by getTime(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED-::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime-::operator()(), and AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()().

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

- airsched/bom/OnDPeriodStruct.hpp

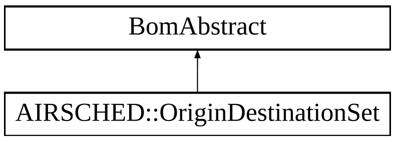
- airsched/bom/OnDPeriodStruct.cpp

23.34 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

23.34.1 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.

23.34.2 Member Typedef Documentation

23.34.2.1 typedef OriginDestinationSetKey AIRSCHED::OriginDestinationSet::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 57 of file OriginDestinationSet.hpp.

23.34.3 Constructor & Destructor Documentation

23.34.3.1 AIRSCHED::OriginDestinationSet::OriginDestinationSet (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 31 of file OriginDestinationSet.cpp.

23.34.3.2 AIRSCHED::OriginDestinationSet::∼OriginDestinationSet ( ) `[protected]`

Destructor.

Definition at line 36 of file OriginDestinationSet.cpp.

23.34.4 Member Function Documentation

23.34.4.1 const Key_T& AIRSCHED::OriginDestinationSet::getKey ( ) const `[inline]`

Get the primary key (destination airport).

Definition at line 65 of file OriginDestinationSet.hpp.

References _key.

23.34.4.2 const stdair::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().

23.34.4.3 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.

23.34.4.4 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.

23.34.4.5 void AIRSCHED::OriginDestinationSet::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 98 of file OriginDestinationSet.hpp.

References toString().

23.34.4.6 void AIRSCHED::OriginDestinationSet::fromStream (std::istream & *ioIn*) [inline]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 107 of file OriginDestinationSet.hpp.

23.34.4.7 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().

23.34.4.8 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().

23.34.4.9 template<class Archive > void AIRSCHED::OriginDestinationSet::serialize (Archive & *ar,* const unsigned int *iFileVersion*)

Serialisation.

Definition at line 62 of file OriginDestinationSet.cpp.

References _key.

23.34.5 Friends And Related Function Documentation

23.34.5.1 friend class stdair::FacBom [friend]

Friend classes.

Definition at line 48 of file OriginDestinationSet.hpp.

23.34.5.2 friend class stdair::FacBomManager [friend]

Definition at line 49 of file OriginDestinationSet.hpp.

23.34.5.3 friend class boost::serialization::access [friend]

Definition at line 50 of file OriginDestinationSet.hpp.

23.34.6 Member Data Documentation

23.34.6.1 Key_T AIRSCHED::OriginDestinationSet::_key [protected]

Primary key (destination airport code).

Definition at line 168 of file OriginDestinationSet.hpp.

Referenced by describeKey(), getDestination(), getKey(), serialize(), and toString().

23.34.6.2 stdair::BomAbstract∗ AIRSCHED::OriginDestinationSet::_parent [protected]

Pointer on the parent (ReachableUniverse) object.

Definition at line 173 of file OriginDestinationSet.hpp.

Referenced by getParent().

23.34.6.3 stdair::HolderMap_T AIRSCHED::OriginDestinationSet::_holderMap [protected]

Map holding the children (SegmentPathPeriod objects).

Definition at line 178 of file OriginDestinationSet.hpp.

Referenced by getHolderMap().

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

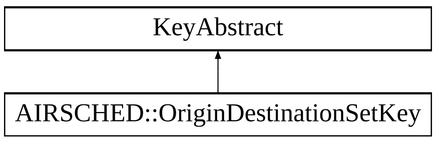
- airsched/bom/OriginDestinationSet.hpp
- airsched/bom/OriginDestinationSet.cpp

23.35 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

23.35.1 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.

23.35.2 Constructor & Destructor Documentation

23.35.2.1 AIRSCHED::OriginDestinationSetKey::OriginDestinationSetKey (const stdair::AirportCode_T & *iDestination*)

Constructor.

Definition at line 26 of file OriginDestinationSetKey.cpp.

23.35.2.2 AIRSCHED::OriginDestinationSetKey::OriginDestinationSetKey (const OriginDestinationSetKey & *iKey*)

Copy constructor.

Definition at line 32 of file OriginDestinationSetKey.cpp.

23.35.2.3 AIRSCHED::OriginDestinationSetKey::∼OriginDestinationSetKey ( )

Destructor.

Definition at line 37 of file OriginDestinationSetKey.cpp.

23.35.3 Member Function Documentation

23.35.3.1 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().

23.35.3.2 void AIRSCHED::OriginDestinationSetKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters

ostream&	the output stream.

Definition at line 41 of file OriginDestinationSetKey.cpp.

References toString().

23.35.3.3 void AIRSCHED::OriginDestinationSetKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters

istream&	the input stream.

Definition at line 46 of file OriginDestinationSetKey.cpp.

23.35.3.4 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::OriginDestination-Set::toString().

23.35.3.5 template<class Archive > void AIRSCHED::OriginDestinationSetKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 72 of file OriginDestinationSetKey.cpp.

23.35.4 Friends And Related Function Documentation

23.35.4.1 friend class boost::serialization::access `[friend]`

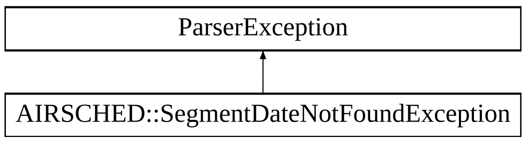
Definition at line 31 of file OriginDestinationSetKey.hpp.

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

- airsched/bom/OriginDestinationSetKey.hpp
- airsched/bom/OriginDestinationSetKey.cpp

23.36 ParserException Class Reference

Inheritance diagram for ParserException:



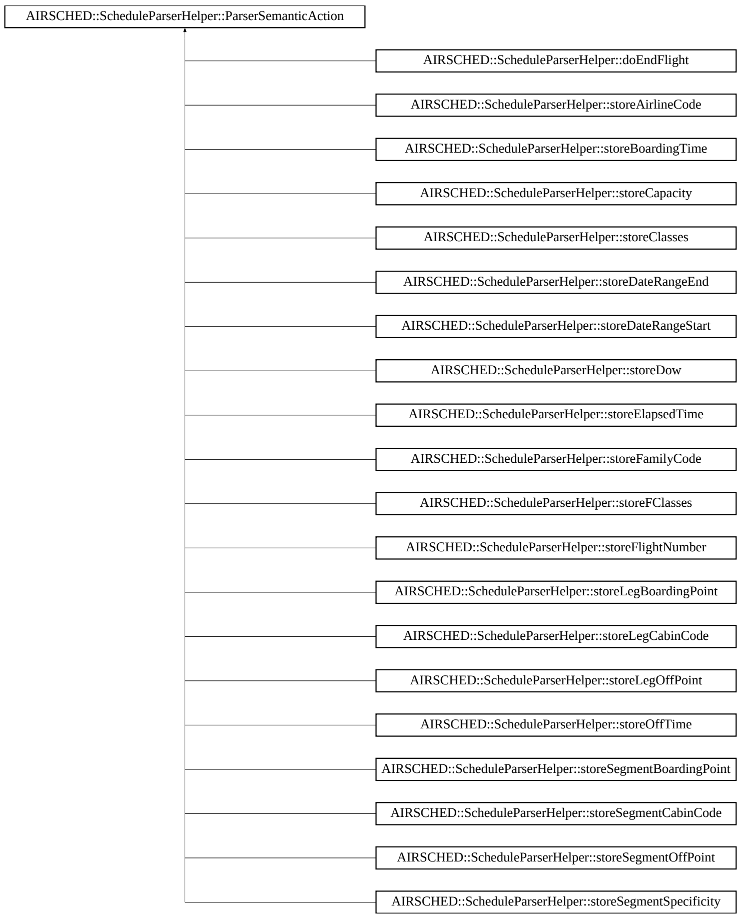
The documentation for this class was generated from the following file:

- airsched/AIRSCHED_Types.hpp

23.37 AIRSCHED::ScheduleParserHelper::ParserSemanticAction Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::ParserSemanticAction:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::doEndFlight
AIRSCHED::ScheduleParserHelper::storeAirlineCode
AIRSCHED::ScheduleParserHelper::storeBoardingTime
AIRSCHED::ScheduleParserHelper::storeCapacity
AIRSCHED::ScheduleParserHelper::storeClasses
AIRSCHED::ScheduleParserHelper::storeDateRangeEnd
AIRSCHED::ScheduleParserHelper::storeDateRangeStart
AIRSCHED::ScheduleParserHelper::storeDow
AIRSCHED::ScheduleParserHelper::storeElapsedTime
AIRSCHED::ScheduleParserHelper::storeFamilyCode
AIRSCHED::ScheduleParserHelper::storeFClasses
AIRSCHED::ScheduleParserHelper::storeFlightNumber
AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint
AIRSCHED::ScheduleParserHelper::storeLegCabinCode
AIRSCHED::ScheduleParserHelper::storeLegOffPoint
AIRSCHED::ScheduleParserHelper::storeOffTime
AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint
AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode
AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint
AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity

Public Member Functions

- ParserSemanticAction (FlightPeriodStruct &)

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.37.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

Definition at line 29 of file ScheduleParserHelper.hpp.

23.37.2 Constructor & Destructor Documentation

23.37.2.1 AIRSCHED::ScheduleParserHelper::ParserSemanticAction::ParserSemanticAction (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 26 of file ScheduleParserHelper.cpp.

23.37.3 Member Data Documentation

23.37.3.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

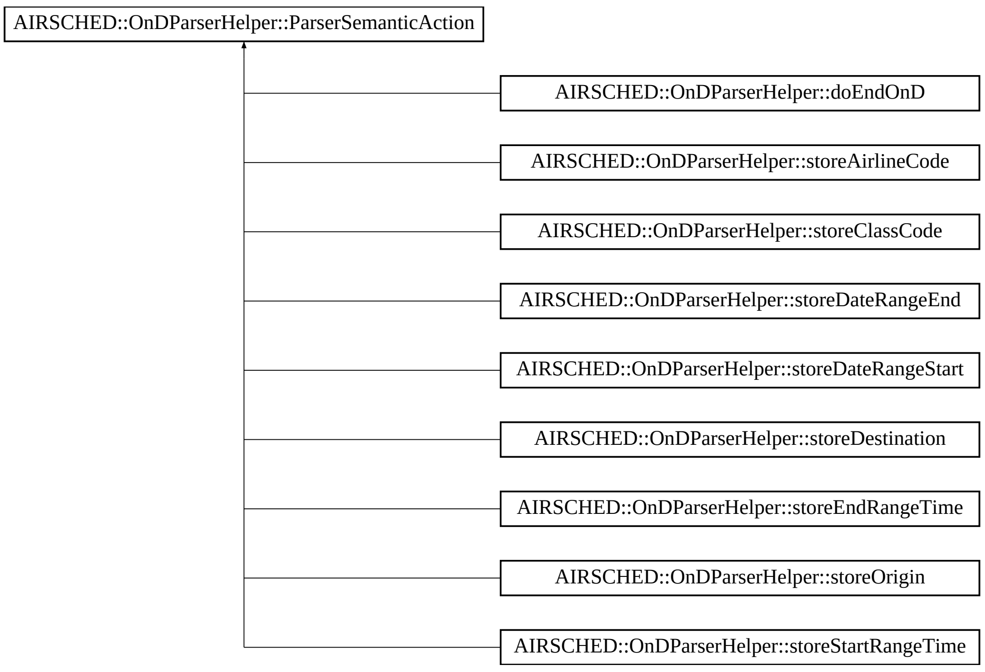
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.38 AIRSCHED::OnDParserHelper::ParserSemanticAction Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::ParserSemanticAction:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::doEndOnD
AIRSCHED::OnDParserHelper::storeAirlineCode
AIRSCHED::OnDParserHelper::storeClassCode
AIRSCHED::OnDParserHelper::storeDateRangeEnd
AIRSCHED::OnDParserHelper::storeDateRangeStart
AIRSCHED::OnDParserHelper::storeDestination
AIRSCHED::OnDParserHelper::storeEndRangeTime
AIRSCHED::OnDParserHelper::storeOrigin
AIRSCHED::OnDParserHelper::storeStartRangeTime

Public Member Functions

- ParserSemanticAction (OnDPeriodStruct &)

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.38.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

Definition at line 34 of file OnDParserHelper.hpp.

23.38.2 Constructor & Destructor Documentation

23.38.2.1 AIRSCHED::OnDParserHelper::ParserSemanticAction::ParserSemanticAction (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 25 of file OnDParserHelper.cpp.

23.38.3 Member Data Documentation

23.38.3.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

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

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.39 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]

23.39.1 Detailed Description

Passenger.

Definition at line 71 of file BookingRequestParser.hpp.

23.39.2 Member Enumeration Documentation

23.39.2.1 enum airsched::Passenger_T::PassengerType_T

Enumerator:

ADULT

CHILD

PET

LAST_VALUE

Definition at line 73 of file BookingRequestParser.hpp.

23.39.3 Constructor & Destructor Documentation

23.39.3.1 airsched::Passenger_T::Passenger_T ( ) [inline]

Constructor.

Definition at line 78 of file BookingRequestParser.hpp.

23.39.4 Member Function Documentation

23.39.4.1 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().

23.39.5 Member Data Documentation

23.39.5.1 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().

23.39.5.2 PassengerType_T airsched::Passenger_T::_type

Definition at line 75 of file BookingRequestParser.hpp.

Referenced by display(), airsched::store_adult_passenger_type::operator()(), airsched::store_child_passenger_-type::operator()(), and airsched::store_pet_passenger_type::operator()().

23.39.5.3 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

23.40 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

23.40.1 Detailed Description

Place.

Definition at line 11 of file BookingRequestParser.hpp.

23.40.2 Constructor & Destructor Documentation

23.40.2.1 airsched::Place_T::Place_T ( ) [inline]

Constructor.

Definition at line 16 of file BookingRequestParser.hpp.

23.40.3 Member Function Documentation

23.40.3.1 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().

23.40.4 Member Data Documentation

23.40.4.1 std::string airsched::Place_T::_name

Definition at line 13 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_place_element::operator()().

23.40.4.2 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:

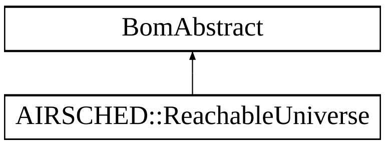
- airsched/batches/BookingRequestParser.hpp

23.41 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

23.41.1 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.

23.41.2 Member Typedef Documentation

23.41.2.1 typedef **ReachableUniverseKey AIRSCHED::ReachableUniverse::Key_T**

Definition allowing to retrieve the associated BOM key type.

Definition at line 55 of file ReachableUniverse.hpp.

23.41.3 Constructor & Destructor Documentation

23.41.3.1 AIRSCHED::ReachableUniverse::ReachableUniverse (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 32 of file ReachableUniverse.cpp.

23.41.3.2 AIRSCHED::ReachableUniverse::∼ReachableUniverse ( ) `[protected]`

Destructor.

Definition at line 37 of file ReachableUniverse.cpp.

23.41.4 Member Function Documentation

23.41.4.1 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.

23.41.4.2 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().

23.41.4.3 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.

23.41.4.4 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.

23.41.4.5 const SegmentPathPeriodListList_T& AIRSCHED::ReachableUniverse::getSegmentPathPeriodListList ( ) const [inline]

Get the vector of SegmentPathPeriodLightList objects.

Definition at line 91 of file ReachableUniverse.hpp.

References _segmentPathPeriodListList.

23.41.4.6 void AIRSCHED::ReachableUniverse::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 103 of file ReachableUniverse.hpp.

References toString().

23.41.4.7 void AIRSCHED::ReachableUniverse::fromStream (std::istream & *ioIn*) [inline]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 112 of file ReachableUniverse.hpp.

23.41.4.8 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().

23.41.4.9 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().

23.41.4.10 template<class Archive > void AIRSCHED::ReachableUniverse::serialize (Archive & *ar,* const unsigned int *iFileVersion*)

Serialisation.

Definition at line 63 of file ReachableUniverse.cpp.

References _key.

23.41.5 Friends And Related Function Documentation

23.41.5.1 friend class stdair::FacBom [friend]

Friend classes.

Definition at line 45 of file ReachableUniverse.hpp.

23.41.5.2 friend class stdair::FacBomManager [friend]

Definition at line 46 of file ReachableUniverse.hpp.

23.41.5.3 friend class SegmentPathGenerator [friend]

Definition at line 47 of file ReachableUniverse.hpp.

23.41.5.4 friend class boost::serialization::access [friend]

Definition at line 48 of file ReachableUniverse.hpp.

23.41.6 Member Data Documentation

23.41.6.1 Key_T AIRSCHED::ReachableUniverse::_key [protected]

Primary key (origin airport code).

Definition at line 174 of file ReachableUniverse.hpp.

Referenced by describeKey(), getKey(), getOrigin(), serialize(), and toString().

23.41.6.2 stdair::BomAbstract* AIRSCHED::ReachableUniverse::_parent [protected]

Pointer on the parent (BomRoot) object.

Definition at line 179 of file ReachableUniverse.hpp.

Referenced by getParent().

23.41.6.3 stdair::HolderMap_T AIRSCHED::ReachableUniverse::_holderMap [protected]

Map holding the children (OriginDestinationSet objects).

Definition at line 184 of file ReachableUniverse.hpp.

Referenced by getHolderMap().

23.41.6.4 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 191 of file ReachableUniverse.hpp.

Referenced by getSegmentPathPeriodListList().

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

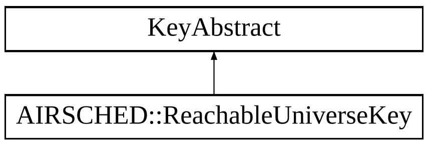
- airsched/bom/ReachableUniverse.hpp
- airsched/bom/ReachableUniverse.cpp

23.42 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

23.42.1 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.

23.42.2 Constructor & Destructor Documentation

23.42.2.1 AIRSCHED::ReachableUniverseKey::ReachableUniverseKey (const stdair::AirportCode_T & *iOrigin*)

Constructor.

Definition at line 32 of file ReachableUniverseKey.cpp.

23.42.2.2 AIRSCHED::ReachableUniverseKey::ReachableUniverseKey (const ReachableUniverseKey & *iKey*)

Copy constructor.

Definition at line 26 of file ReachableUniverseKey.cpp.

23.42.2.3 AIRSCHED::ReachableUniverseKey::∼ReachableUniverseKey ( )

Destructor.

Definition at line 37 of file ReachableUniverseKey.cpp.

23.42.3 Member Function Documentation

23.42.3.1 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().

23.42.3.2 void AIRSCHED::ReachableUniverseKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters

ostream&	the output stream.

Definition at line 41 of file ReachableUniverseKey.cpp.

References toString().

23.42.3.3 void AIRSCHED::ReachableUniverseKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters

istream&	the input stream.

Definition at line 46 of file ReachableUniverseKey.cpp.

23.42.3.4 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().

23.42.3.5 template<class Archive > void AIRSCHED::ReachableUniverseKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 72 of file ReachableUniverseKey.cpp.

23.42.4 Friends And Related Function Documentation

23.42.4.1 friend class boost::serialization::access [friend]

Definition at line 34 of file ReachableUniverseKey.hpp.

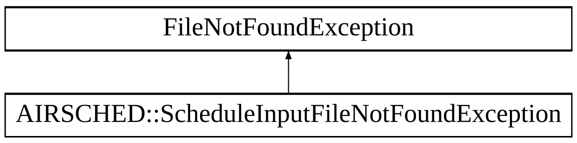
The documentation for this struct was generated from the following files:

- airsched/bom/ReachableUniverseKey.hpp
- airsched/bom/ReachableUniverseKey.cpp

23.43 AIRSCHED::ScheduleInputFileNotFoundException Class Reference

`#include <airsched/AIRSCHED_Types.hpp>`

Inheritance diagram for AIRSCHED::ScheduleInputFileNotFoundException:



Public Member Functions

- ScheduleInputFileNotFoundException (const std::string &iWhat)

23.43.1 Detailed Description

The schedule input file cannot be retrieved.

Definition at line 47 of file AIRSCHED_Types.hpp.

23.43.2 Constructor & Destructor Documentation

23.43.2.1 AIRSCHED::ScheduleInputFileNotFoundException::ScheduleInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 53 of file AIRSCHED_Types.hpp.

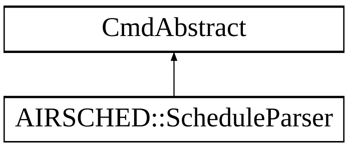
The documentation for this class was generated from the following file:

- airsched/AIRSCHED_Types.hpp

23.44 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 &)

23.44.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 21 of file ScheduleParser.hpp.

23.44.2 Member Function Documentation

23.44.2.1 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

const	stdair::Filename_T& The file-name of the CSV-formatted schedule input file.
stdair::Bom-Root&	Root of the BOM tree.

Definition at line 18 of file ScheduleParser.cpp.

References AIRSCHED::SegmentPathGenerator::createSegmentPathNetwork(), and AIRSCHED::FlightPeriodFile-Parser::generateInventories().

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

23.45 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

23.45.1 Detailed Description

Search string.

Definition at line 94 of file BookingRequestParser.hpp.

23.45.2 Constructor & Destructor Documentation

23.45.2.1 airsched::SearchString_T::SearchString_T ( ) [inline]

Constructor.

Definition at line 102 of file BookingRequestParser.hpp.

23.45.3 Member Function Documentation

23.45.3.1 void airsched::SearchString_T::display ( ) const [inline]

Definition at line 105 of file BookingRequestParser.hpp.

References _airlineList, _dateList, _passengerList, _placeList, _tmpPlace, airsched::Place_T::display(), airsched::-Date_T::display(), airsched::Airline_T::display(), and airsched::Passenger_T::display().

23.45.4 Member Data Documentation

23.45.4.1 PlaceList_T airsched::SearchString_T::_placeList

Definition at line 96 of file BookingRequestParser.hpp.

Referenced by display().

23.45.4.2 DateList_T airsched::SearchString_T::_dateList

Definition at line 97 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_date::operator()().

23.45.4.3 AirlineList_T airsched::SearchString_T::_airlineList

Definition at line 98 of file BookingRequestParser.hpp.

Referenced by display(), airsched::store_airline_code::operator()(), and airsched::store_airline_name::operator()().

23.45.4.4 PassengerList_T airsched::SearchString_T::_passengerList

Definition at line 99 of file BookingRequestParser.hpp.

Referenced by display(), airsched::store_adult_passenger_type::operator()(), airsched::store_child_passenger_-type::operator()(), and airsched::store_pet_passenger_type::operator()().

23.45.4.5 Place_T airsched::SearchString_T::_tmpPlace

Definition at line 137 of file BookingRequestParser.hpp.

Referenced by display(), and airsched::store_place_element::operator()().

23.45.4.6 Date_T airsched::SearchString_T::_tmpDate

Definition at line 138 of file BookingRequestParser.hpp.

Referenced by airsched::store_date::operator()().

23.45.4.7 Airline_T airsched::SearchString_T::_tmpAirline

Definition at line 139 of file BookingRequestParser.hpp.

Referenced by airsched::store_airline_sign::operator()(), airsched::store_airline_code::operator()(), and airsched::store_airline_name::operator()().

23.45.4.8 Passenger_T airsched::SearchString_T::_tmpPassenger

Definition at line 140 of file BookingRequestParser.hpp.

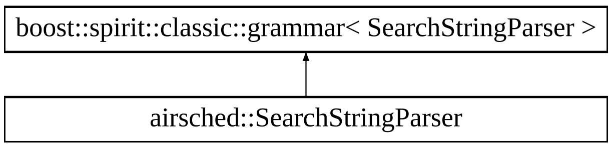
Referenced by airsched::store_passenger_number::operator()(), airsched::store_adult_passenger_type::operator()(), airsched::store_child_passenger_type::operator()(), and airsched::store_pet_passenger_type::operator()().

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

- airsched/batches/BookingRequestParser.hpp

23.46 airsched::SearchStringParser Struct Reference

Inheritance diagram for airsched::SearchStringParser:


boost::spirit::classic::grammar< SearchStringParser >
airsched::SearchStringParser

Classes

- struct definition

Public Member Functions

- SearchStringParser (SearchString_T &ioSearchString)

Public Attributes

- SearchString_T & _searchString

23.46.1 Detailed Description

Grammar for the search string parser.

Definition at line 251 of file BookingRequestParser.cpp.

23.46.2 Constructor & Destructor Documentation

23.46.2.1 airsched::SearchStringParser::SearchStringParser (SearchString_T & *ioSearchString*) [inline]

Definition at line 254 of file BookingRequestParser.cpp.

23.46.3 Member Data Documentation

23.46.3.1 SearchString_T& airsched::SearchStringParser::_searchString

Definition at line 369 of file BookingRequestParser.cpp.

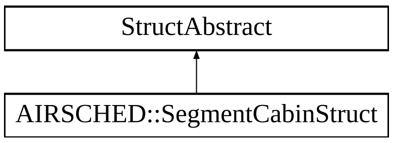
The documentation for this struct was generated from the following file:

- airsched/batches/BookingRequestParser.cpp

23.47 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

23.47.1 Detailed Description

Utility Structure for the parsing of SegmentCabin details.

Definition at line 24 of file SegmentCabinStruct.hpp.

23.47.2 Member Function Documentation

23.47.2.1 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.

23.47.2.2 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().

23.47.3 Member Data Documentation

23.47.3.1 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()().

23.47.3.2 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()().

23.47.3.3 stdair::FamilyCode_T AIRSCHED::SegmentCabinStruct::_itFamilyCode

Definition at line 28 of file SegmentCabinStruct.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), and AIRSCHED::ScheduleParserHelper::storeFClasses::operator()().

23.47.3.4 FareFamilyStructList_T AIRSCHED::SegmentCabinStruct::_fareFamilies

Definition at line 29 of file SegmentCabinStruct.hpp.

Referenced by AIRSCHED::FlightPeriodStruct::addFareFamily().

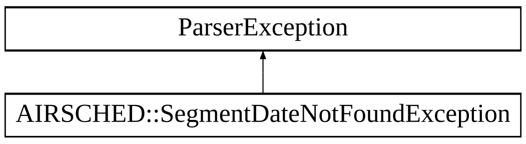
The documentation for this struct was generated from the following files:

- airsched/bom/SegmentCabinStruct.hpp
- airsched/bom/SegmentCabinStruct.cpp

23.48 AIRSCHED::SegmentDateNotFoundException Class Reference

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

Inheritance diagram for AIRSCHED::SegmentDateNotFoundException:



Public Member Functions

- SegmentDateNotFoundException (const std::string &iWhat)

23.48.1 Detailed Description

Specific exception when some BOM objects can not be found within the schedule.

Definition at line 23 of file AIRSCHED_Types.hpp.

23.48.2 Constructor & Destructor Documentation

23.48.2.1 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:

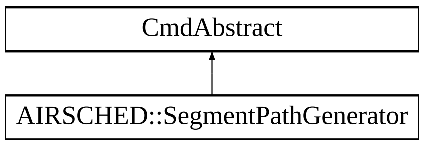
- airsched/AIRSCHED_Types.hpp

23.49 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 &)

23.49.1 Detailed Description

Class handling the generation / instantiation of the network BOM.

Definition at line 34 of file SegmentPathGenerator.hpp.

23.49.2 Member Function Documentation

23.49.2.1 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::generateInventories().

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

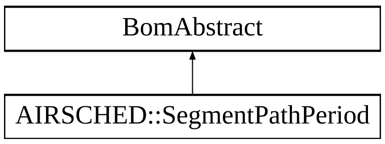
- airsched/command/SegmentPathGenerator.hpp
- airsched/command/SegmentPathGenerator.cpp

23.50 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

23.50.1 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.

23.50.2 Member Typedef Documentation

23.50.2.1 typedef SegmentPathPeriodKey AIRSCHED::SegmentPathPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 52 of file SegmentPathPeriod.hpp.

23.50.3 Constructor & Destructor Documentation

23.50.3.1 AIRSCHED::SegmentPathPeriod::SegmentPathPeriod (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 43 of file SegmentPathPeriod.cpp.

23.50.3.2 AIRSCHED::SegmentPathPeriod::∼SegmentPathPeriod ( ) [protected]

Destructor.

Definition at line 48 of file SegmentPathPeriod.cpp.

23.50.4 Member Function Documentation

23.50.4.1 const Key_T& AIRSCHED::SegmentPathPeriod::getKey ( ) const [inline]

Get the primary key (destination airport).

Definition at line 60 of file SegmentPathPeriod.hpp.

References _key.

23.50.4.2 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.

23.50.4.3 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().

23.50.4.4 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().

23.50.4.5 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().

23.50.4.6 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().

23.50.4.7 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().

23.50.4.8 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().

23.50.4.9 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.

23.50.4.10 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 91 of file SegmentPathPeriod.cpp.

Referenced by connectWithAnotherSegment(), and getDestination().

23.50.4.11 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 109 of file SegmentPathPeriod.cpp.

Referenced by connectWithAnotherSegment().

23.50.4.12 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 127 of file SegmentPathPeriod.cpp.

References getLastSegmentPeriod().

23.50.4.13 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 163 of file SegmentPathPeriod.cpp.

References checkCircle(), getBoardingDateOffsetList(), getBoardingTime(), getDeparturePeriod(), getElapsedTime(), getFirstSegmentPeriod(), getLastSegmentPeriod(), getNbOfSegments(), AIRSCHED::SegmentPathPeriodKey::setBoardingDateOffsetList(), AIRSCHED::SegmentPathPeriodKey::setBoardingTime(), AIRSCHED::SegmentPathPeriodKey::setElapsedTime(), and AIRSCHED::SegmentPathPeriodKey::setPeriod().

23.50.4.14 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 289 of file SegmentPathPeriod.cpp.

Referenced by connectWithAnotherSegment().

23.50.4.15 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 135 of file SegmentPathPeriod.cpp.

23.50.4.16 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 308 of file SegmentPathPeriod.cpp.

References getDeparturePeriod().

23.50.4.17 void AIRSCHED::SegmentPathPeriod::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 176 of file SegmentPathPeriod.hpp.

References toString().

23.50.4.18 void AIRSCHED::SegmentPathPeriod::fromStream (std::istream & *ioIn*) [inline]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 185 of file SegmentPathPeriod.hpp.

23.50.4.19 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().

23.50.4.20 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().

23.50.4.21 template<class Archive > void AIRSCHED::SegmentPathPeriod::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 74 of file SegmentPathPeriod.cpp.

References _key.

23.50.5 Friends And Related Function Documentation

23.50.5.1 friend class stdair::FacBom [friend]

Friend classes.

Definition at line 43 of file SegmentPathPeriod.hpp.

23.50.5.2 friend class stdair::FacBomManager [friend]

Definition at line 44 of file SegmentPathPeriod.hpp.

23.50.5.3 friend class boost::serialization::access [friend]

Definition at line 45 of file SegmentPathPeriod.hpp.

23.50.6 Member Data Documentation

23.50.6.1 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 249 of file SegmentPathPeriod.hpp.

Referenced by describeKey(), getBoardingDateOffsetList(), getBoardingTime(), getDeparturePeriod(), getElapsed-Time(), getKey(), getNbOfAirlines(), getNbOfSegments(), serialize(), and toString().

23.50.6.2 stdair::BomAbstract∗ AIRSCHED::SegmentPathPeriod::_parent [protected]

Pointer on the parent (OriginDestinationSet) object.

Definition at line 254 of file SegmentPathPeriod.hpp.

Referenced by getParent().

23.50.6.3 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 262 of file SegmentPathPeriod.hpp.

Referenced by getHolderMap().

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

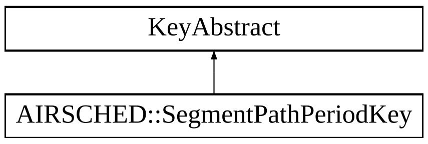
- airsched/bom/SegmentPathPeriod.hpp
- airsched/bom/SegmentPathPeriod.cpp

23.51 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

23.51.1 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.

23.51.2 Constructor & Destructor Documentation

23.51.2.1 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 40 of file SegmentPathPeriodKey.cpp.

23.51.2.2 AIRSCHED::SegmentPathPeriodKey::SegmentPathPeriodKey ( )

Default constructor.

Definition at line 22 of file SegmentPathPeriodKey.cpp.

23.51.2.3 AIRSCHED::SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey & *iSPPK*)

Copy constructor.

Definition at line 30 of file SegmentPathPeriodKey.cpp.

23.51.2.4 AIRSCHED::SegmentPathPeriodKey::∼SegmentPathPeriodKey ( )

Destructor.

Definition at line 53 of file SegmentPathPeriodKey.cpp.

23.51.3 Member Function Documentation

23.51.3.1 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().

23.51.3.2 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().

23.51.3.3 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().

23.51.3.4 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().

23.51.3.5 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().

23.51.3.6 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().

23.51.3.7 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().

23.51.3.8 void AIRSCHED::SegmentPathPeriodKey::setBoardingDateOffsetList (const DateOffsetList_T & *iList*) [inline]

Definition at line 115 of file SegmentPathPeriodKey.hpp.

Referenced by AIRSCHED::SegmentPathPeriod::connectWithAnotherSegment().

23.51.3.9 void AIRSCHED::SegmentPathPeriodKey::setNbOfAirlines (const stdair::NbOfAirlines_T & *iNbOfAirlines*) [inline]

Set the number of airlines.

Definition at line 120 of file SegmentPathPeriodKey.hpp.

23.51.3.10 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().

23.51.3.11 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().

23.51.3.12 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.

23.51.3.13 void AIRSCHED::SegmentPathPeriodKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters

ostream&	the output stream.

Definition at line 57 of file SegmentPathPeriodKey.cpp.

References toString().

23.51.3.14 void AIRSCHED::SegmentPathPeriodKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters

istream&	the input stream.

Definition at line 62 of file SegmentPathPeriodKey.cpp.

23.51.3.15 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 66 of file SegmentPathPeriodKey.cpp.

Referenced by AIRSCHED::SegmentPathPeriod::describeKey(), toStream(), and AIRSCHED::SegmentPathPeriod::toString().

23.51.3.16 template<class Archive > void AIRSCHED::SegmentPathPeriodKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 98 of file SegmentPathPeriodKey.cpp.

23.51.4 Friends And Related Function Documentation

23.51.4.1 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:

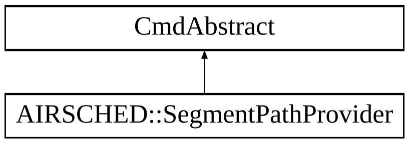
- airsched/bom/SegmentPathPeriodKey.hpp
- airsched/bom/SegmentPathPeriodKey.cpp

23.52 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

23.52.1 Detailed Description

Class building the travel solutions from airline schedules.

Definition at line 27 of file SegmentPathProvider.hpp.

23.52.2 Friends And Related Function Documentation

23.52.2.1 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

23.53 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 &)

23.53.1 Detailed Description

Class representing the actual business functions for an airline segment-period.

Definition at line 19 of file SegmentPeriodHelper.hpp.

23.53.2 Member Function Documentation

23.53.2.1 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.

23.53.2.2 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.

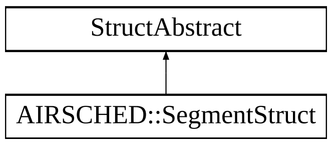
The documentation for this class was generated from the following files:

- airsched/bom/SegmentPeriodHelper.hpp
- airsched/bom/SegmentPeriodHelper.cpp

23.54 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

23.54.1 Detailed Description

Utility Structure for the parsing of Segment structures.

Definition at line 24 of file SegmentStruct.hpp.

23.54.2 Member Function Documentation

23.54.2.1 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.

23.54.2.2 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().

23.54.3 Member Data Documentation

23.54.3.1 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::storeSegmentBoardingPoint::operator()().

23.54.3.2 stdair::Date_T AIRSCHED::SegmentStruct::_boardingDate

Definition at line 27 of file SegmentStruct.hpp.

23.54.3.3 stdair::Duration_T AIRSCHED::SegmentStruct::_boardingTime

Definition at line 28 of file SegmentStruct.hpp.

Referenced by describe().

23.54.3.4 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::storeSegmentOffPoint::operator()().

23.54.3.5 stdair::Date_T AIRSCHED::SegmentStruct::_offDate

Definition at line 30 of file SegmentStruct.hpp.

23.54.3.6 stdair::Duration_T AIRSCHED::SegmentStruct::_offTime

Definition at line 31 of file SegmentStruct.hpp.

Referenced by describe().

23.54.3.7 stdair::Duration_T AIRSCHED::SegmentStruct::_elapsed

Definition at line 32 of file SegmentStruct.hpp.

Referenced by describe().

23.54.3.8 SegmentCabinStructList_T AIRSCHED::SegmentStruct::_cabinList

Definition at line 33 of file SegmentStruct.hpp.

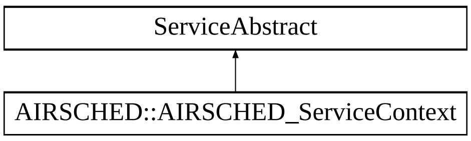
Referenced by AIRSCHED::FlightPeriodStruct::addFareFamily(), AIRSCHED::FlightPeriodStruct::addSegmentCabin(), describe(), and AIRSCHED::SegmentPeriodHelper::fill().

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

- airsched/bom/SegmentStruct.hpp
- airsched/bom/SegmentStruct.cpp

23.55 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



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

- airsched/service/AIRSCHED_ServiceContext.hpp

23.56 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 ()

23.56.1 Detailed Description

Base class for the Service layer.

Definition at line 14 of file ServiceAbstract.hpp.

23.56.2 Constructor & Destructor Documentation

23.56.2.1 virtual AIRSCHED::ServiceAbstract::∼ServiceAbstract ( ) [inline], [virtual]

Destructor.

Definition at line 18 of file ServiceAbstract.hpp.

23.56.2.2 AIRSCHED::ServiceAbstract::ServiceAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 30 of file ServiceAbstract.hpp.

23.56.3 Member Function Documentation

23.56.3.1 virtual void AIRSCHED::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 22 of file ServiceAbstract.hpp.

23.56.3.2 virtual void AIRSCHED::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 26 of file ServiceAbstract.hpp.

Referenced by operator>>().

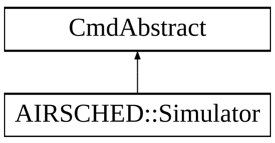
The documentation for this class was generated from the following file:

- airsched/service/ServiceAbstract.hpp

23.57 AIRSCHED::Simulator Class Reference

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

Inheritance diagram for AIRSCHED::Simulator:



Static Public Member Functions

- static void simulate (stdair::BomRoot &)

23.57.1 Detailed Description

Class implementing a small simulation, which uses the Airline Schedule.

Definition at line 18 of file Simulator.hpp.

23.57.2 Member Function Documentation

23.57.2.1 void AIRSCHED::Simulator::simulate (stdair::BomRoot & *ioBomRoot*) [static]

Perform a small simulation, which uses the Airline Schedule.

Parameters

stdair::Bom-Root&	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

23.58 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

23.58.1 Detailed Description

Store the parsed passenger type.

Definition at line 145 of file BookingRequestParser.cpp.

23.58.2 Constructor & Destructor Documentation

23.58.2.1 airsched::store_adult_passenger_type::store_adult_passenger_type (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 147 of file BookingRequestParser.cpp.

23.58.3 Member Function Documentation

23.58.3.1 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.

23.58.4 Member Data Documentation

23.58.4.1 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

23.59 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

23.59.1 Detailed Description

Store the parsed airline code.

Definition at line 92 of file BookingRequestParser.cpp.

23.59.2 Constructor & Destructor Documentation

23.59.2.1 airsched::store_airline_code::store_airline_code (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 94 of file BookingRequestParser.cpp.

23.59.3 Member Function Documentation

23.59.3.1 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.

23.59.4 Member Data Documentation

23.59.4.1 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

23.60 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

23.60.1 Detailed Description

Store the parsed airline name.

Definition at line 111 of file BookingRequestParser.cpp.

23.60.2 Constructor & Destructor Documentation

23.60.2.1 airsched::store_airline_name::store_airline_name (**SearchString_T** & *ioSearchString*) [inline]

Constructor.

Definition at line 113 of file BookingRequestParser.cpp.

23.60.3 Member Function Documentation

23.60.3.1 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.

23.60.4 Member Data Documentation

23.60.4.1 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

23.61 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

23.61.1 Detailed Description

Store the airline sign (+/-).

Definition at line 77 of file BookingRequestParser.cpp.

23.61.2 Constructor & Destructor Documentation

23.61.2.1 airsched::store_airline_sign::store_airline_sign (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 79 of file BookingRequestParser.cpp.

23.61.3 Member Function Documentation

23.61.3.1 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.

23.61.4 Member Data Documentation

23.61.4.1 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

23.62 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

23.62.1 Detailed Description

Store the parsed passenger type.

Definition at line 164 of file BookingRequestParser.cpp.

23.62.2 Constructor & Destructor Documentation

23.62.2.1 airsched::store_child_passenger_type::store_child_passenger_type (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 166 of file BookingRequestParser.cpp.

23.62.3 Member Function Documentation

23.62.3.1 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.

23.62.4 Member Data Documentation

23.62.4.1 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

23.63 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

23.63.1 Detailed Description

Store a parsed date.

Definition at line 58 of file BookingRequestParser.cpp.

23.63.2 Constructor & Destructor Documentation

23.63.2.1 airsched::store_date::store_date (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 60 of file BookingRequestParser.cpp.

23.63.3 Member Function Documentation

23.63.3.1 void airsched::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().

23.63.4 Member Data Documentation

23.63.4.1 SearchString_T& airsched::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

23.64 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

23.64.1 Detailed Description

Store the parsed number of passengers.

Definition at line 130 of file BookingRequestParser.cpp.

23.64.2 Constructor & Destructor Documentation

23.64.2.1 airsched::store_passenger_number::store_passenger_number (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 132 of file BookingRequestParser.cpp.

23.64.3 Member Function Documentation

23.64.3.1 void airsched::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.

23.64.4 Member Data Documentation

23.64.4.1 SearchString_T& airsched::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

23.65 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

23.65.1 Detailed Description

Store the parsed passenger type.

Definition at line 183 of file BookingRequestParser.cpp.

23.65.2 Constructor & Destructor Documentation

23.65.2.1 airsched::store_pet_passenger_type::store_pet_passenger_type (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 185 of file BookingRequestParser.cpp.

23.65.3 Member Function Documentation

23.65.3.1 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.

23.65.4 Member Data Documentation

23.65.4.1 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

23.66 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

23.66.1 Detailed Description

Store the parsed place element.

Definition at line 37 of file BookingRequestParser.cpp.

23.66.2 Constructor & Destructor Documentation

23.66.2.1 airsched::store_place_element::store_place_element (SearchString_T & *ioSearchString*) [inline]

Constructor.

Definition at line 39 of file BookingRequestParser.cpp.

23.66.3 Member Function Documentation

23.66.3.1 void airsched::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.

23.66.4 Member Data Documentation

23.66.4.1 SearchString_T& airsched::store_place_element::_searchString

Definition at line 54 of file BookingRequestParser.cpp.

Referenced by operator()().

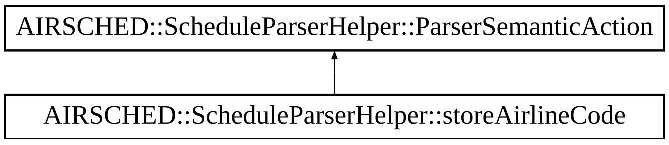
The documentation for this struct was generated from the following file:

- airsched/batches/BookingRequestParser.cpp

23.67 AIRSCHED::ScheduleParserHelper::storeAirlineCode Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeAirlineCode:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeAirlineCode

Public Member Functions

- storeAirlineCode (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.67.1 Detailed Description

Store the parsed airline code.

Definition at line 37 of file ScheduleParserHelper.hpp.

23.67.2 Constructor & Destructor Documentation

23.67.2.1 AIRSCHED::ScheduleParserHelper::storeAirlineCode::storeAirlineCode (**FlightPeriodStruct &** *ioFlightPeriod*)

Actor Constructor.

Definition at line 32 of file ScheduleParserHelper.cpp.

23.67.3 Member Function Documentation

23.67.3.1 void AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator() (**iterator_t** *iStr,* **iterator_t** *iStrEnd*) const

Actor Function (functor).

Definition at line 37 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_airlineCode, AIRSCHED::ScheduleParserHelper::ParserSemantic-Action::_flightPeriod, and AIRSCHED::FlightPeriodStruct::_legList.

23.67.4 Member Data Documentation

23.67.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod `[inherited]`

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::-ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRange-End::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper-

::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

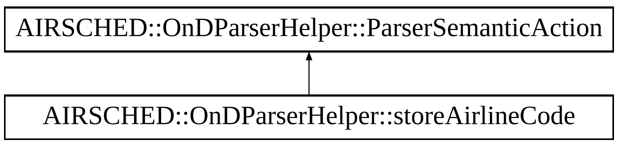
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.68 AIRSCHED::OnDParserHelper::storeAirlineCode Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeAirlineCode:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeAirlineCode

Public Member Functions

- storeAirlineCode (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.68.1 Detailed Description

Store the parsed airline code.

Definition at line 90 of file OnDParserHelper.hpp.

23.68.2 Constructor & Destructor Documentation

23.68.2.1 AIRSCHED::OnDParserHelper::storeAirlineCode::storeAirlineCode (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 139 of file OnDParserHelper.cpp.

23.68.3 Member Function Documentation

23.68.3.1 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.

23.68.4 Member Data Documentation

23.68.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

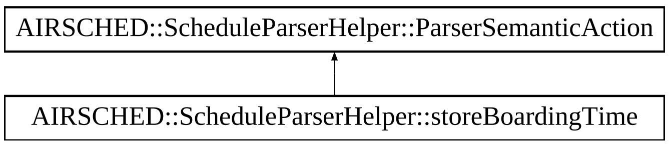
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.69 AIRSCHED::ScheduleParserHelper::storeBoardingTime Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeBoardingTime:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeBoardingTime

Public Member Functions

- storeBoardingTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.69.1 Detailed Description

Store the boarding time.

Definition at line 93 of file ScheduleParserHelper.hpp.

23.69.2 Constructor & Destructor Documentation

23.69.2.1 AIRSCHED::ScheduleParserHelper::storeBoardingTime::storeBoardingTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 155 of file ScheduleParserHelper.cpp.

23.69.3 Member Function Documentation

23.69.3.1 void AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 160 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().

23.69.4 Member Data Documentation

23.69.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper-::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCH-ED::ScheduleParserHelper::storeLegOffPoint::operator()(), operator()(), AIRSCHED::ScheduleParserHelper-::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED-::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity-::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::Schedule-ParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegment-OffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHE-D::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode-::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParser-Helper::doEndFlight::operator()().

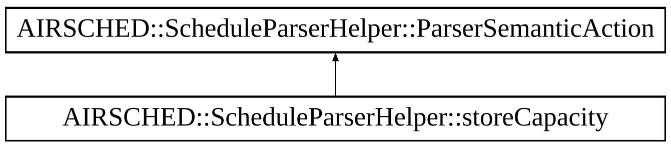
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.70 AIRSCHED::ScheduleParserHelper::storeCapacity Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeCapacity:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeCapacity

Public Member Functions

- storeCapacity (FlightPeriodStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.70.1 Detailed Description

Store the parsed capacity.

Definition at line 125 of file ScheduleParserHelper.hpp.

23.70.2 Constructor & Destructor Documentation

23.70.2.1 AIRSCHED::ScheduleParserHelper::storeCapacity::storeCapacity (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 227 of file ScheduleParserHelper.cpp.

23.70.3 Member Function Documentation

23.70.3.1 void AIRSCHED::ScheduleParserHelper::storeCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 232 of file ScheduleParserHelper.cpp.

References AIRSCHED::LegStruct::_cabinList, AIRSCHED::LegCabinStruct::_capacity, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itLeg, and AIRSCHED::FlightPeriodStruct::_itLegCabin.

23.70.4 Member Data Documentation

23.70.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

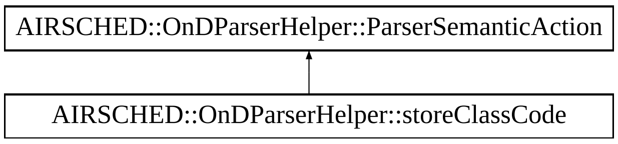
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.71 AIRSCHED::OnDParserHelper::storeClassCode Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeClassCode:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeClassCode

Public Member Functions

- storeClassCode (OnDPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.71.1 Detailed Description

Store the parsed class code.

Definition at line 98 of file OnDParserHelper.hpp.

23.71.2 Constructor & Destructor Documentation

23.71.2.1 AIRSCHED::OnDParserHelper::storeClassCode::storeClassCode (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 172 of file OnDParserHelper.cpp.

23.71.3 Member Function Documentation

23.71.3.1 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.

23.71.4 Member Data Documentation

23.71.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime-

::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

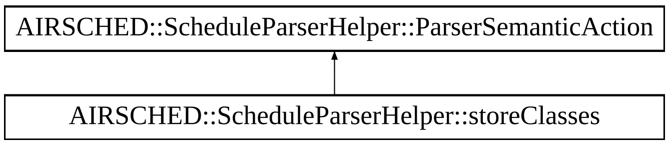
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.72 AIRSCHED::ScheduleParserHelper::storeClasses Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeClasses:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeClasses

Public Member Functions

- storeClasses (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.72.1 Detailed Description

Store the parsed list of class codes.

Definition at line 168 of file ScheduleParserHelper.hpp.

23.72.2 Constructor & Destructor Documentation

23.72.2.1 AIRSCHED::ScheduleParserHelper::storeClasses::storeClasses (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 309 of file ScheduleParserHelper.cpp.

23.72.3 Member Function Documentation

23.72.3.1 void AIRSCHED::ScheduleParserHelper::storeClasses::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 314 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().

23.72.4 Member Data Documentation

23.72.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper-::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED-::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime-::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper-::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRS-CHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegment-Specificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRS-CHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::store-SegmentCabinCode::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEnd-Flight::operator()().

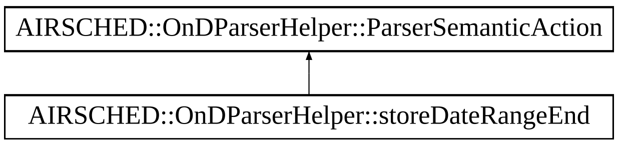
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.73 AIRSCHED::OnDParserHelper::storeDateRangeEnd Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeDateRangeEnd:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeDateRangeEnd

Public Member Functions

- storeDateRangeEnd (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.73.1 Detailed Description

Store the end of the date range.

Definition at line 66 of file OnDParserHelper.hpp.

23.73.2 Constructor & Destructor Documentation

23.73.2.1 AIRSCHED::OnDParserHelper::storeDateRangeEnd::storeDateRangeEnd (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 83 of file OnDParserHelper.cpp.

23.73.3 Member Function Documentation

23.73.3.1 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::getDate().

23.73.4 Member Data Documentation

23.73.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

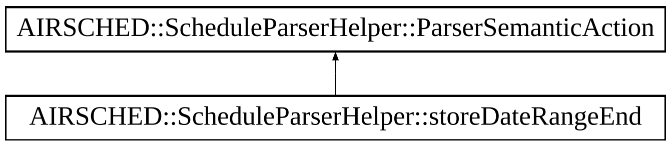
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.74 AIRSCHED::ScheduleParserHelper::storeDateRangeEnd Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeDateRangeEnd:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeDateRangeEnd

Public Member Functions

- storeDateRangeEnd (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.74.1 Detailed Description

Store the end of the date range.

Definition at line 61 of file ScheduleParserHelper.hpp.

23.74.2 Constructor & Destructor Documentation

23.74.2.1 AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::storeDateRangeEnd (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 75 of file ScheduleParserHelper.cpp.

23.74.3 Member Function Documentation

23.74.3.1 void AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 80 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_dateRange, AIRSCHED::FlightPeriodStruct::_dateRangeEnd, AIRSCHED::FlightPeriodStruct::_dateRangeStart, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itSeconds, and AIRSCHED::FlightPeriodStruct::getDate().

23.74.4 Member Data Documentation

23.74.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

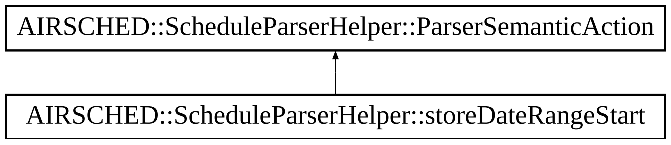
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.75 AIRSCHED::ScheduleParserHelper::storeDateRangeStart Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeDateRangeStart:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeDateRangeStart

Public Member Functions

- storeDateRangeStart (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.75.1 Detailed Description

Store the start of the date range.

Definition at line 53 of file ScheduleParserHelper.hpp.

23.75.2 Constructor & Destructor Documentation

23.75.2.1 AIRSCHED::ScheduleParserHelper::storeDateRangeStart::storeDateRangeStart (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 60 of file ScheduleParserHelper.cpp.

23.75.3 Member Function Documentation

23.75.3.1 void AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 65 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_dateRangeStart, AIRSCHED::ScheduleParserHelper::Parser-SemanticAction::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itSeconds, and AIRSCHED::FlightPeriodStruct-::getDate().

23.75.4 Member Data Documentation

23.75.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

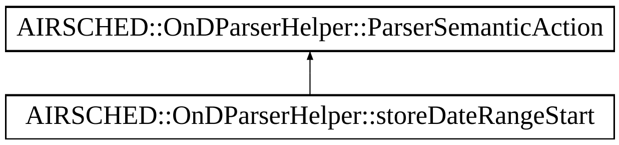
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.76 AIRSCHED::OnDParserHelper::storeDateRangeStart Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeDateRangeStart:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeDateRangeStart

Public Member Functions

- storeDateRangeStart (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.76.1 Detailed Description

Store the start of the date range.

Definition at line 58 of file OnDParserHelper.hpp.

23.76.2 Constructor & Destructor Documentation

23.76.2.1 AIRSCHED::OnDParserHelper::storeDateRangeStart::storeDateRangeStart (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 66 of file OnDParserHelper.cpp.

23.76.3 Member Function Documentation

23.76.3.1 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::getDate().

23.76.4 Member Data Documentation

23.76.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

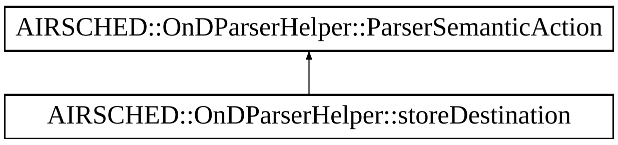
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.77 AIRSCHED::OnDParserHelper::storeDestination Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeDestination:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeDestination

Public Member Functions

- storeDestination (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.77.1 Detailed Description

Store the parsed destination.

Definition at line 50 of file OnDParserHelper.hpp.

23.77.2 Constructor & Destructor Documentation

23.77.2.1 AIRSCHED::OnDParserHelper::storeDestination::storeDestination (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 50 of file OnDParserHelper.cpp.

23.77.3 Member Function Documentation

23.77.3.1 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.

23.77.4 Member Data Documentation

23.77.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

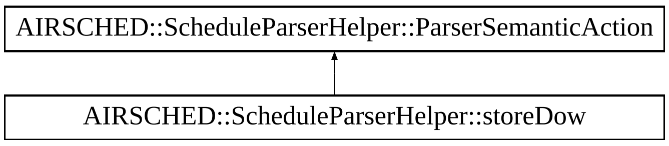
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.78 AIRSCHED::ScheduleParserHelper::storeDow Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeDow:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeDow

Public Member Functions

- storeDow (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.78.1 Detailed Description

Store the DOW (day of the Week).

Definition at line 69 of file ScheduleParserHelper.hpp.

23.78.2 Constructor & Destructor Documentation

23.78.2.1 AIRSCHED::ScheduleParserHelper::storeDow::storeDow (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 98 of file ScheduleParserHelper.cpp.

23.78.3 Member Function Documentation

23.78.3.1 void AIRSCHED::ScheduleParserHelper::storeDow::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 103 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_dow, and AIRSCHED::ScheduleParserHelper::ParserSemantic-Action::_flightPeriod.

23.78.4 Member Data Documentation

23.78.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), operator()(), AIRSCHED::ScheduleParser-Helper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper-::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED-::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity-::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::Schedule-ParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegment-OffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHE-D::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode-::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParser-Helper::doEndFlight::operator()().

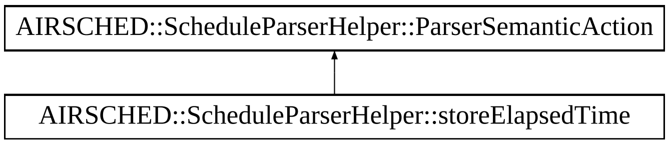
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.79 AIRSCHED::ScheduleParserHelper::storeElapsedTime Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeElapsedTime:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeElapsedTime

Public Member Functions

- storeElapsedTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.79.1 Detailed Description

Store the elapsed time.

Definition at line 109 of file ScheduleParserHelper.hpp.

23.79.2 Constructor & Destructor Documentation

23.79.2.1 AIRSCHED::ScheduleParserHelper::storeElapsedTime::storeElapsedTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 194 of file ScheduleParserHelper.cpp.

23.79.3 Member Function Documentation

23.79.3.1 void AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 199 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_dateOffset, AIRSCHED::LegStruct::_elapsed, AIRSCHED::Schedule-ParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itLeg, AIRSCHED::Flight-PeriodStruct::_itSeconds, AIRSCHED::LegStruct::_offDateOffset, and AIRSCHED::FlightPeriodStruct::getTime().

23.79.4 Member Data Documentation

23.79.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper-::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHE-D::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoarding-Time::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), operator()(), AIRSCHED-::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity-::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::Schedule-

ParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegment-OffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHE-D::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode-::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParser-Helper::doEndFlight::operator()().

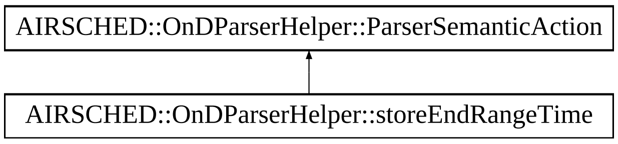
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.80 AIRSCHED::OnDParserHelper::storeEndRangeTime Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeEndRangeTime:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeEndRangeTime

Public Member Functions

- storeEndRangeTime (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.80.1 Detailed Description

Store the end range time.

Definition at line 82 of file OnDParserHelper.hpp.

23.80.2 Constructor & Destructor Documentation

23.80.2.1 AIRSCHED::OnDParserHelper::storeEndRangeTime::storeEndRangeTime (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 124 of file OnDParserHelper.cpp.

23.80.3 Member Function Documentation

23.80.3.1 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().

23.80.4 Member Data Documentation

23.80.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

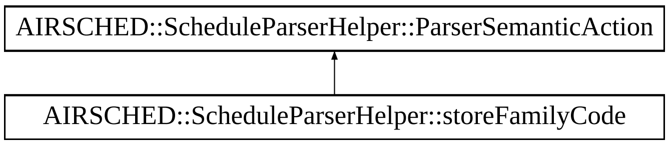
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.81 AIRSCHED::ScheduleParserHelper::storeFamilyCode Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFamilyCode:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeFamilyCode

Public Member Functions

- storeFamilyCode (FlightPeriodStruct &)
- void operator() (int iCode) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.81.1 Detailed Description

Store the parsed family code.

Definition at line 176 of file ScheduleParserHelper.hpp.

23.81.2 Constructor & Destructor Documentation

23.81.2.1 AIRSCHED::ScheduleParserHelper::storeFamilyCode::storeFamilyCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 334 of file ScheduleParserHelper.cpp.

23.81.3 Member Function Documentation

23.81.3.1 void AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

Definition at line 339 of file ScheduleParserHelper.cpp.

References AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::SegmentCabinStruct::_itFamilyCode, and AIRSCHED::FlightPeriodStruct::_itSegmentCabin.

23.81.4 Member Data Documentation

23.81.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

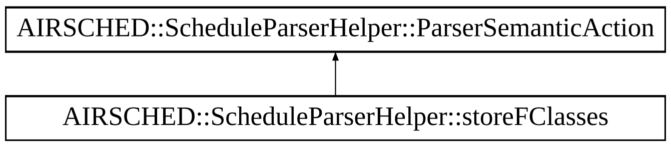
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.82 AIRSCHED::ScheduleParserHelper::storeFClasses Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFClasses:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeFClasses

Public Member Functions

- storeFClasses (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.82.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 184 of file ScheduleParserHelper.hpp.

23.82.2 Constructor & Destructor Documentation

23.82.2.1 AIRSCHED::ScheduleParserHelper::storeFClasses::storeFClasses (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 347 of file ScheduleParserHelper.cpp.

23.82.3 Member Function Documentation

23.82.3.1 void AIRSCHED::ScheduleParserHelper::storeFClasses::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 352 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::SegmentCabinStruct::_itFamilyCode, AIRSCHED::FlightPeriodStruct::_itSegment, AIRSCHED::FlightPeriodStruct::_itSegmentCabin, and AIRSCHED::FlightPeriodStruct::addFareFamily().

23.82.4 Member Data Documentation

23.82.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

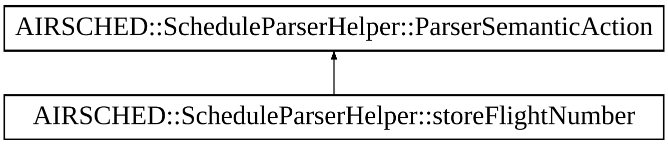
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.83 AIRSCHED::ScheduleParserHelper::storeFlightNumber Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeFlightNumber:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeFlightNumber

Public Member Functions

- storeFlightNumber (FlightPeriodStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.83.1 Detailed Description

Store the parsed flight number.

Definition at line 45 of file ScheduleParserHelper.hpp.

23.83.2 Constructor & Destructor Documentation

23.83.2.1 AIRSCHED::ScheduleParserHelper::storeFlightNumber::storeFlightNumber (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 49 of file ScheduleParserHelper.cpp.

23.83.3 Member Function Documentation

23.83.3.1 void AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 54 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_flightNumber, and AIRSCHED::ScheduleParserHelper::Parser-SemanticAction::_flightPeriod.

23.83.4 Member Data Documentation

23.83.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), operator()(), AIRSCHED::-ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRange-End::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper-::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSC-HED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime-::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParser-Helper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRS-CHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::store-SegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), A-

IRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper-::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED-::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight-::operator()().

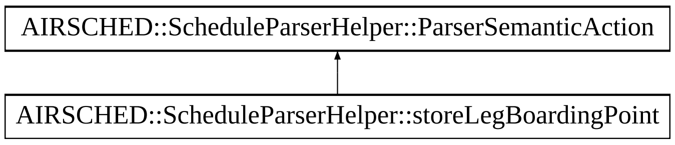
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.84 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint

Public Member Functions

- storeLegBoardingPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.84.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 77 of file ScheduleParserHelper.hpp.

23.84.2 Constructor & Destructor Documentation

23.84.2.1 AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::storeLegBoardingPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 110 of file ScheduleParserHelper.cpp.

23.84.3 Member Function Documentation

23.84.3.1 void AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 115 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().

23.84.4 Member Data Documentation

23.84.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

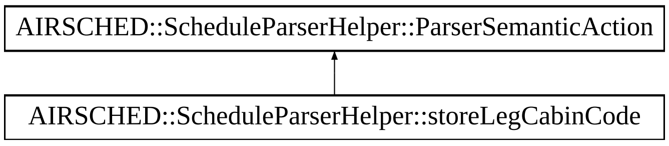
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.85 AIRSCHED::ScheduleParserHelper::storeLegCabinCode Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegCabinCode:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeLegCabinCode

Public Member Functions

- storeLegCabinCode (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.85.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 117 of file ScheduleParserHelper.hpp.

23.85.2 Constructor & Destructor Documentation

23.85.2.1 AIRSCHED::ScheduleParserHelper::storeLegCabinCode::storeLegCabinCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 215 of file ScheduleParserHelper.cpp.

23.85.3 Member Function Documentation

23.85.3.1 void AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 220 of file ScheduleParserHelper.cpp.

References AIRSCHED::LegCabinStruct::_cabinCode, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRSCHED::FlightPeriodStruct::_itLegCabin.

23.85.4 Member Data Documentation

23.85.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

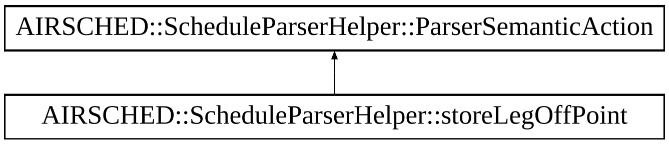
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.86 AIRSCHED::ScheduleParserHelper::storeLegOffPoint Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeLegOffPoint:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeLegOffPoint

Public Member Functions

- storeLegOffPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.86.1 Detailed Description

Store the parsed leg off point.

Definition at line 85 of file ScheduleParserHelper.hpp.

23.86.2 Constructor & Destructor Documentation

23.86.2.1 AIRSCHED::ScheduleParserHelper::storeLegOffPoint::storeLegOffPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 139 of file ScheduleParserHelper.cpp.

23.86.3 Member Function Documentation

23.86.3.1 void AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 144 of file ScheduleParserHelper.cpp.

References AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::FlightPeriod-Struct::_itLeg, AIRSCHED::LegStruct::_offPoint, and AIRSCHED::FlightPeriodStruct::addAirport().

23.86.4 Member Data Documentation

23.86.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::store-Dow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), operator()(), AIRSC-HED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime-::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParser-Helper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRS-CHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::store-SegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), A-

IRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

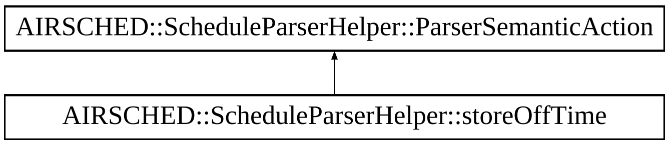
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.87 AIRSCHED::ScheduleParserHelper::storeOffTime Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeOffTime:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeOffTime

Public Member Functions

- storeOffTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.87.1 Detailed Description

Store the off time.

Definition at line 101 of file ScheduleParserHelper.hpp.

23.87.2 Constructor & Destructor Documentation

23.87.2.1 AIRSCHED::ScheduleParserHelper::storeOffTime::storeOffTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 173 of file ScheduleParserHelper.cpp.

23.87.3 Member Function Documentation

23.87.3.1 void AIRSCHED::ScheduleParserHelper::storeOffTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 178 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().

23.87.4 Member Data Documentation

23.87.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

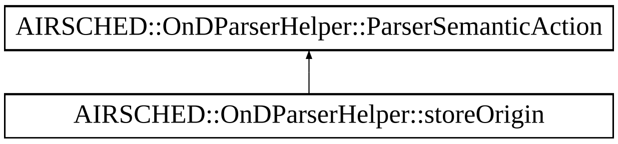
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.88 AIRSCHED::OnDParserHelper::storeOrigin Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeOrigin:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeOrigin

Public Member Functions

- storeOrigin (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.88.1 Detailed Description

Store the parsed origin.

Definition at line 42 of file OnDParserHelper.hpp.

23.88.2 Constructor & Destructor Documentation

23.88.2.1 AIRSCHED::OnDParserHelper::storeOrigin::storeOrigin (OnDPeriodStruct & *ioOnDPeriod*)

Actor Constructor.

Definition at line 30 of file OnDParserHelper.cpp.

23.88.3 Member Function Documentation

23.88.3.1 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.

23.88.4 Member Data Documentation

23.88.4.1 OnDPeriodStruct& AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod [inherited]

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by operator()(), AIRSCHED::OnDParserHelper::storeDestination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::OnDParserHelper::storeStartRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeEndRangeTime::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

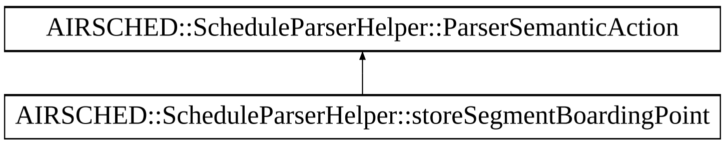
The documentation for this struct was generated from the following files:

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.89 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint

Public Member Functions

- storeSegmentBoardingPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.89.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 144 of file ScheduleParserHelper.hpp.

23.89.2 Constructor & Destructor Documentation

23.89.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 272 of file ScheduleParserHelper.cpp.

23.89.3 Member Function Documentation

23.89.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 277 of file ScheduleParserHelper.cpp.

References AIRSCHED::SegmentStruct::_boardingPoint, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRSCHED::FlightPeriodStruct::_itSegment.

23.89.4 Member Data Documentation

23.89.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

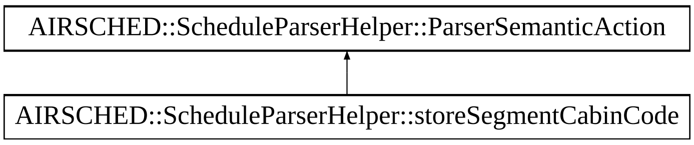
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.90 AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode

Public Member Functions

- storeSegmentCabinCode (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.90.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 160 of file ScheduleParserHelper.hpp.

23.90.2 Constructor & Destructor Documentation

23.90.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::storeSegmentCabinCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 298 of file ScheduleParserHelper.cpp.

23.90.3 Member Function Documentation

23.90.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 303 of file ScheduleParserHelper.cpp.

References AIRSCHED::SegmentCabinStruct::_cabinCode, AIRSCHED::ScheduleParserHelper::ParserSemantic-Action::_flightPeriod, and AIRSCHED::FlightPeriodStruct::_itSegmentCabin.

23.90.4 Member Data Documentation

23.90.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParser-Helper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

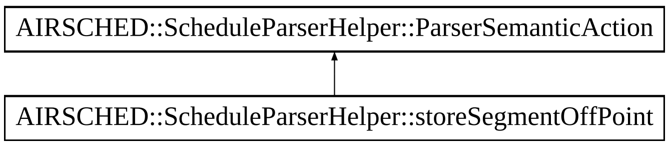
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.91 AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint

Public Member Functions

- storeSegmentOffPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.91.1 Detailed Description

Store the parsed segment off point.

Definition at line 152 of file ScheduleParserHelper.hpp.

23.91.2 Constructor & Destructor Documentation

23.91.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::storeSegmentOffPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 285 of file ScheduleParserHelper.cpp.

23.91.3 Member Function Documentation

23.91.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 290 of file ScheduleParserHelper.cpp.

References AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRSCHED::FlightPeriodStruct::_itSegment, and AIRSCHED::SegmentStruct::_offPoint.

23.91.4 Member Data Documentation

23.91.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

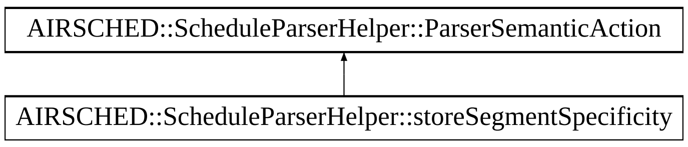
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.92 AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity Struct Reference

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

Inheritance diagram for AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity:


AIRSCHED::ScheduleParserHelper::ParserSemanticAction
AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity

Public Member Functions

- storeSegmentSpecificity (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

23.92.1 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 136 of file ScheduleParserHelper.hpp.

23.92.2 Constructor & Destructor Documentation

23.92.2.1 AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::storeSegmentSpecificity (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 246 of file ScheduleParserHelper.cpp.

23.92.3 Member Function Documentation

23.92.3.1 void AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 251 of file ScheduleParserHelper.cpp.

References AIRSCHED::FlightPeriodStruct::_airportList, AIRSCHED::FlightPeriodStruct::_airportOrderedList, AIRSCHED::FlightPeriodStruct::_areSegmentDefinitionsSpecific, AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRSCHED::FlightPeriodStruct::buildSegments().

23.92.4 Member Data Documentation

23.92.4.1 FlightPeriodStruct& AIRSCHED::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRSCHED::ScheduleParserHelper::storeAirlineCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFlightNumber::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRSCHED::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRSCHED::ScheduleParserHelper::storeDow::operator()(), AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeBoardingTime::operator()(), AIRSCHED::ScheduleParserHelper::storeOffTime::operator()(), AIRSCHED::ScheduleParserHelper::storeElapsedTime::operator()(), AIRSCHED::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeCapacity::operator()(), operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRSCHED::ScheduleParserHelper::storeClasses::operator()(), AIRSCHED::ScheduleParserHelper::storeFamilyCode::operator()(), AIRSCHED::ScheduleParserHelper::storeFClasses::operator()(), and AIRSCHED::ScheduleParserHelper::doEndFlight::operator()().

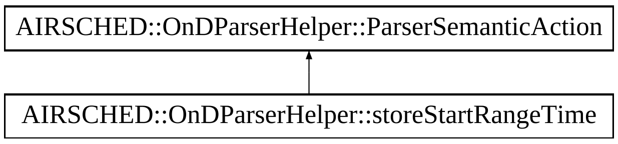
The documentation for this struct was generated from the following files:

- airsched/command/ScheduleParserHelper.hpp
- airsched/command/ScheduleParserHelper.cpp

23.93 AIRSCHED::OnDParserHelper::storeStartRangeTime Struct Reference

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

Inheritance diagram for AIRSCHED::OnDParserHelper::storeStartRangeTime:


AIRSCHED::OnDParserHelper::ParserSemanticAction
AIRSCHED::OnDParserHelper::storeStartRangeTime

Public Member Functions

- storeStartRangeTime (OnDPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- OnDPeriodStruct & _onDPeriod

23.93.1 Detailed Description

Store the start range time.

Definition at line 74 of file OnDParserHelper.hpp.

23.93.2 Constructor & Destructor Documentation

23.93.2.1 AIRSCHED::OnDParserHelper::storeStartRangeTime::storeStartRangeTime (**OnDPeriodStruct** & *ioOnDPeriod*)

Actor Constructor.

Definition at line 109 of file OnDParserHelper.cpp.

23.93.3 Member Function Documentation

23.93.3.1 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().

23.93.4 Member Data Documentation

23.93.4.1 **OnDPeriodStruct&** AIRSCHED::OnDParserHelper::ParserSemanticAction::_onDPeriod `[inherited]`

Actor Context.

Definition at line 38 of file OnDParserHelper.hpp.

Referenced by AIRSCHED::OnDParserHelper::storeOrigin::operator()(), AIRSCHED::OnDParserHelper::store-Destination::operator()(), AIRSCHED::OnDParserHelper::storeDateRangeStart::operator()(), AIRSCHED::OnD-ParserHelper::storeDateRangeEnd::operator()(), operator()(), AIRSCHED::OnDParserHelper::storeEndRange-

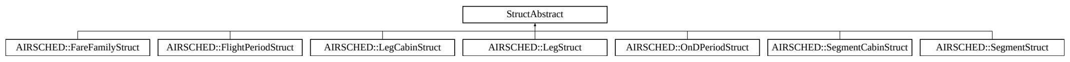
Time::operator()(), AIRSCHED::OnDParserHelper::storeAirlineCode::operator()(), AIRSCHED::OnDParserHelper-::storeClassCode::operator()(), and AIRSCHED::OnDParserHelper::doEndOnD::operator()().

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

- airsched/command/OnDParserHelper.hpp
- airsched/command/OnDParserHelper.cpp

23.94 StructAbstract Class Reference

Inheritance diagram for StructAbstract:

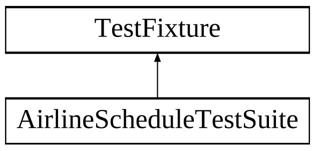

StructAbstract
AIRSCHED::FareFamilyStruct
AIRSCHED::FlightPeriodStruct
AIRSCHED::LegCabinStruct
AIRSCHED::LegStruct
AIRSCHED::OnDPeriodStruct
AIRSCHED::SegmentCabinStruct
AIRSCHED::SegmentStruct

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

- airsched/bom/SegmentStruct.hpp

23.95 TestFixture Class Reference

Inheritance diagram for TestFixture:



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

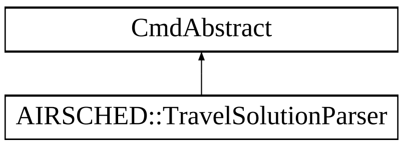
- test/airsched/AirlineScheduleTestSuite.hpp

23.96 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 &)

23.96.1 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.

23.96.2 Member Function Documentation

23.96.2.1 bool AIRSCHED::TravelSolutionParser::parseInputFileAndBuildBom (const stdair::Filename_T &) [static]

Parse the input values from a CSV-formatted travel solution file.

Parameters

const	std::string& iInputFileName Travel solution file to be parsed.

Returns

bool Whether or not the parsing was successful.

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

24 File Documentation

24.1 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

24.2 AIRSCHED_Service.hpp

```
#ifndef __AIRSCHED_SVC_AIRSCHED_SERVICE_HPP
#define __AIRSCHED_SVC_AIRSCHED_SERVICE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>

namespace stdair {
  class STDAIR_Service;
  struct BasLogParams;
  struct BasDBParams;
  struct BookingRequestStruct;
  struct TravelSolutionStruct;
}

namespace AIRSCHED {

  class AIRSCHED_ServiceContext;


  class AIRSCHED_Service {
  public:
    // ////////////////// Constructors and Destructors //////////////////
    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();


  public:
    // /////////// Business Methods /////////////
    void buildSampleBom();

    void buildSegmentPathList (stdair::TravelSolutionList_T&,
                               const stdair::BookingRequestStruct&);

    void simulate();


  public:
    // //////////////// Export support methods /////////////////
    std::string jsonExport (const stdair::AirlineCode_T&,
                            const stdair::FlightNumber_T&,
                            const stdair::Date_T& iDepartureDate) const;


  public:
    // //////////////// Display support methods /////////////////
    std::string csvDisplay() const;

    std::string csvDisplay (const stdair::AirlineCode_T&,
                            const stdair::FlightNumber_T&,
                            const stdair::Date_T& iDepartureDate) const;


  private:
    // /////// Construction and Destruction helper methods ///////
    AIRSCHED_Service();

    AIRSCHED_Service (const AIRSCHED_Service&);

    stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
                                                   const stdair::BasDBParams&);

    stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&);

    void addStdAirService (stdair::STDAIR_ServicePtr_T,
                           const bool iOwnStdairService);

    void initServiceContext();

    void initAirschedService();

    void finalise();
  private:
    // ///////// Service Context /////////
    AIRSCHED_ServiceContext* _airschedServiceContext;
  };
}
#endif // __AIRSCHED_SVC_AIRSCHED_SERVICE_HPP
```

24.3 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

24.4 AIRSCHED_Types.hpp

```
#ifndef __AIRSCHED_AIRSCHED_TYPES_HPP
#define __AIRSCHED_AIRSCHED_TYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// Boost
#include <boost/shared_ptr.hpp>
// StdAir
#include <stdair/stdair_exceptions.hpp>

namespace AIRSCHED {

  // Forward declarations
  class AIRSCHED_Service;


  // ///////// Exceptions ///////////
  class SegmentDateNotFoundException : public
stdair::ParserException {
  public:
    SegmentDateNotFoundException (const std::string
& iWhat)
      : stdair::ParserException (iWhat) {}
  };

  class OnDInputFileNotFoundException : public
stdair::FileNotFoundException {
  public:
    OnDInputFileNotFoundException (const
std::string& iWhat)
      : stdair::FileNotFoundException (iWhat) {}
  };

  class ScheduleInputFileNotFoundException
    : public stdair::FileNotFoundException {
  public:
    ScheduleInputFileNotFoundException (const
 std::string& iWhat)
      : stdair::FileNotFoundException (iWhat) {}
  };


  // //////// Type definitions specific to AirSched /////////
  typedef boost::shared_ptr<AIRSCHED_Service> AIRSCHED_ServicePtr_T
;

}
#endif // __AIRSCHED_AIRSCHED_TYPES_HPP
```

24.5 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

24.6 BasConst.cpp

```
#include <airsched/basic/BasConst_General.hpp
>
#include <airsched/basic/BasConst_AIRSCHED_Service.hpp
>

namespace AIRSCHED {

  const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
 = 100000;

}
```

24.7 airsched/basic/BasConst_AIRSCHED_Service.hpp File Reference

Namespaces

- namespace AIRSCHED

24.8 BasConst_AIRSCHED_Service.hpp

```
#ifndef __AIRSCHED_BAS_BASCONST_AIRSCHED_SERVICE_HPP
#define __AIRSCHED_BAS_BASCONST_AIRSCHED_SERVICE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////

namespace AIRSCHED {

}
#endif // __AIRSCHED_BAS_BASCONST_AIRSCHED_SERVICE_HPP
```

24.9 airsched/basic/BasConst_General.hpp File Reference

Namespaces

- namespace AIRSCHED

24.10 BasConst_General.hpp

```
#ifndef __AIRSCHED_BAS_BASCONST_GENERAL_HPP
#define __AIRSCHED_BAS_BASCONST_GENERAL_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////

namespace AIRSCHED {

  extern const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
;

}
#endif // __AIRSCHED_BAS_BASCONST_GENERAL_HPP
```

24.11 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

24.12 BasParserTypes.hpp

```
#ifndef __AIRSCHED_BAS_BASCOMPARSERTYPES_HPP
#define __AIRSCHED_BAS_BASCOMPARSERTYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost
//#define BOOST_SPIRIT_DEBUG
#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>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  //
  //  Definition of Basic Types
  //
  // //////////////////////////////////////////////////////////////////////
  // For a file, the parsing unit is the character (char). For a string,
  // it is a "char const *".
  // typedef char const* iterator_t;
  typedef char char_t;

  // The types of iterator, scanner and rule are then derived from
  // the parsing unit.
  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;

  // //////////////////////////////////////////////////////////////////////
  //
  //  Parser related types
  //
  // //////////////////////////////////////////////////////////////////////
  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;

}
#endif // __AIRSCHED_BAS_BASCOMPARSERTYPES_HPP
```

24.13 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 &ioIsBuiltin, bool &ioReadBookingRequestFromCmdLine, stdair::Filename_T &ioInputFilename, std::string &ioLogFilename, std::string &ioBookingRequestString)
- stdair::BookingRequestStruct parseBookingRequest (const std::string &iRequestOption)
- int main (int argc, char ∗argv[])

Variables

- const bool K_AIRSCHED_DEFAULT_BUILT_IN_INPUT = false
- const bool K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE = false
- const int K_AIRSCHED_EARLY_RETURN_STATUS = 99

24.13.1 Typedef Documentation

24.13.1.1 typedef std::vector<std::string> WordList_T

Definition at line 24 of file airsched.cpp.

24.13.2 Function Documentation

24.13.2.1 const std::string K_AIRSCHED_DEFAULT_LOG_FILENAME ("airsched.log")

Default name and location for the log file.

Referenced by readConfiguration().

24.13.2.2 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().

24.13.2.3 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 seached against the AirSched network.

Referenced by main().

24.13.2.4 std::string createStringFromWordList (const WordList_T & *iWordList*)

Definition at line 59 of file airsched.cpp.

Referenced by readConfiguration().

24.13.2.5 template<class T > std::ostream& operator<< (std::ostream & *os,* const std::vector< T > & *v*)

Definition at line 77 of file airsched.cpp.

24.13.2.6 int readConfiguration (int *argc,* char ∗ *argv[ ],* bool & *ioIsBuiltin,* bool & *ioReadBookingRequestFromCmdLine,* stdair::Filename_T & *ioInputFilename,* std::string & *ioLogFilename,* std::string & *ioBookingRequestString*)

Read and parse the command line options.

Definition at line 87 of file airsched.cpp.

References createStringFromWordList(), K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE, K_AIRSCHE-D_DEFAULT_BUILT_IN_INPUT, K_AIRSCHED_DEFAULT_INPUT_FILENAME(), K_AIRSCHED_DEFAULT_LO-G_FILENAME(), and K_AIRSCHED_EARLY_RETURN_STATUS.

Referenced by main().

24.13.2.7 stdair::BookingRequestStruct parseBookingRequest (const std::string & *iRequestOption*)

Definition at line 230 of file airsched.cpp.

Referenced by main().

24.13.2.8 int main (int *argc*, char ∗ *argv[]*)

Definition at line 335 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().

24.13.3 Variable Documentation

24.13.3.1 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.

Referenced by readConfiguration().

24.13.3.2 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().

24.13.3.3 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().

24.14 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/AIRSCHED_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_AIRSCHED_DEFAULT_LOG_FILENAME
        ("airsched.log");
00032
00036 const std::string K_AIRSCHED_DEFAULT_INPUT_FILENAME
        (STDAIR_SAMPLE_DIR
00037                                                        "/schedule03.csv");
00038
```

```
const bool K_AIRSCHED_DEFAULT_BUILT_IN_INPUT =
 false;

const bool K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE
 = false;

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) {
  std::ostringstream oStr;

  unsigned short idx = iWordList.size();
  for (WordList_T::const_iterator itWord = iWordList.begin();
       itWord != iWordList.end(); ++itWord, --idx) {
    const std::string& lWord = *itWord;
    oStr << lWord;
    if (idx > 1) {
      oStr << " ";
    }
  }

  return oStr.str();
}

// ///////// Parsing of Options & Configuration /////////
// A helper function to simplify the main part.
template<class T> std::ostream& operator<< (std::ostream& os,
                                            const std::vector<T>& v) {
  std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
  return os;
}

const int K_AIRSCHED_EARLY_RETURN_STATUS = 99;

int readConfiguration (int argc, char* argv[],
                       bool& ioIsBuiltin, bool& ioReadBookingRequestFromCmdLine
,
                       stdair::Filename_T& ioInputFilename,
                       std::string& ioLogFilename,
                       std::string& ioBookingRequestString) {

  // Default for the built-in input
  ioIsBuiltin = K_AIRSCHED_DEFAULT_BUILT_IN_INPUT
;

  // Default for the booking request mode (whether it is read from
 command-line)
  ioReadBookingRequestFromCmdLine = K_AIRSCHED_DEFAULT_BOOKING_REQUEST_MODE
;

  //
  WordList_T lWordList;

  // Declare a group of options that will be allowed only on command line
  boost::program_options::options_description generic ("Generic options");
  generic.add_options()
    ("prefix", "print installation prefix")
    ("version,v", "print version string")
    ("help,h", "produce help message");

  // Declare a group of options that will be allowed both on command
  // line and in config file
  boost::program_options::options_description config ("Configuration");
  config.add_options()
    ("builtin,b",
     "The sample BOM tree can be either built-in or parsed from input files. In
 that latter case, the -i/--input option must be specified as well")
    ("input,i",
     boost::program_options::value< std::string >(&ioInputFilename)->
default_value(K_AIRSCHED_DEFAULT_INPUT_FILENAME),
     "(CSV) input file specifying the schedule (flight-period) entries")
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->
default_value(K_AIRSCHED_DEFAULT_LOG_FILENAME),
     "Filename for the logs")
    ("read_booking_request,r",
     "Indicates that a booking request is given as a command-line option. That
 latter must then be given with the -b/--bkg_req option")
    ("bkg_req,q",
     boost::program_options::value< WordList_T >(&lWordList)->multitoken(),
     "Booking request word list (e.g. 'NCE BKK NCE 2007-04-21 2007-04-21
 10:00:00 C 1 DF RO 5 NONE 10:0:0 2000.0 20.0'), which should be located at the end of
 the command line (otherwise, the other options would be interpreted as part of
that booking request word list)")
    ;

  // Hidden options, will be allowed both on command line and
  // in config file, but will not be shown to the user.
  boost::program_options::options_description hidden ("Hidden options");
  hidden.add_options()
    ("copyright",
     boost::program_options::value< std::vector<std::string> >(),
     "Show the copyright (license)");

  boost::program_options::options_description cmdline_options;
  cmdline_options.add(generic).add(config).add(hidden);

  boost::program_options::options_description config_file_options;
  config_file_options.add(config).add(hidden);

  boost::program_options::options_description visible ("Allowed options");
  visible.add(generic).add(config);

  boost::program_options::positional_options_description p;
  p.add ("copyright", -1);

  boost::program_options::variables_map vm;
  boost::program_options::
    store (boost::program_options::command_line_parser (argc, argv).
           options (cmdline_options).positional(p).run(), vm);

  std::ifstream ifs ("airsched.cfg");
  boost::program_options::store (parse_config_file (ifs, config_file_options),
                                 vm);
  boost::program_options::notify (vm);

  if (vm.count ("help")) {
    std::cout << visible << std::endl;
    return K_AIRSCHED_EARLY_RETURN_STATUS;
  }

  if (vm.count ("version")) {
    std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
    return K_AIRSCHED_EARLY_RETURN_STATUS;
  }

  if (vm.count ("prefix")) {
    std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
    return K_AIRSCHED_EARLY_RETURN_STATUS;
  }

  if (vm.count ("builtin")) {
    ioIsBuiltin = true;
  }
  const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
  std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

  //
  std::ostringstream oErrorMessageStr;
  oErrorMessageStr << "Either the -b/--builtin option, or the -i/--input option
"
                   << " must be specified";

  if (ioIsBuiltin == false) {
    if (vm.count ("input")) {
      ioInputFilename = vm["input"].as< std::string >();
      std::cout << "Input filename is: " << ioInputFilename << std::endl;

    } else {
      // The built-in option is not selected. However, no schedule input file
      // is specified
      std::cerr << oErrorMessageStr.str() << std::endl;
    }
  }

  //
  if (vm.count ("read_booking_request")) {
    ioReadBookingRequestFromCmdLine = true;
  }
  const std::string readBookingRequestFromCmdLineStr =
    (ioReadBookingRequestFromCmdLine == true)?"yes":"no";
  std::cout << "A booking request is to be given as command-line option? "
            << readBookingRequestFromCmdLineStr << std::endl;

  if (ioReadBookingRequestFromCmdLine == true) {

    if (lWordList.empty() == true) {
      std::cerr << "When the --read_booking_request/-r option is given, "
                << "a query must also be provided (with the --bkg_req/-b "
                << "option at the end of the command-line)" << std::endl;
      return K_AIRSCHED_EARLY_RETURN_STATUS;
    }

    // Rebuild the booking request query string
    ioBookingRequestString = createStringFromWordList (
lWordList);
    std::cout << "The booking request string is: " << ioBookingRequestString
              << std::endl;
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}

// //////////////////////////////////////////////////////////////
stdair::BookingRequestStruct
parseBookingRequest (const std::string& iRequestOption) {
  typedef boost::tokenizer<boost::char_separator<char> > tokenizer;
  boost::char_separator<char> sep(" -:");

  tokenizer tokens (iRequestOption, sep);

  // Origin (e.g., "NCE")
  tokenizer::iterator tok_iter = tokens.begin();
  assert (tok_iter != tokens.end());
  const stdair::AirportCode_T iOrigin (*tok_iter);

  // Destination (e.g., "BKK")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::AirportCode_T iDestination (*tok_iter);

  // POS (e.g., "NCE")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::AirportCode_T iPOS (*tok_iter);

  // Preferred departure date (e.g., "2007-04-21")
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lDepDateYear = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lDepDateMonth = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lDepDateDay = boost::lexical_cast<short> (*tok_iter);
  const stdair::Date_T iDepartureDate(lDepDateYear, lDepDateMonth, lDepDateDay)
;

  // Request date (e.g., "2007-03-21")
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqDateYear = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqDateMonth = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqDateDay = boost::lexical_cast<short> (*tok_iter);
  const stdair::Date_T iRequestDate (lReqDateYear, lReqDateMonth, lReqDateDay);

  // Request time (e.g., "08:34:23")
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqTimeHours = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqTimeMinutes = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lReqTimeSeconds = boost::lexical_cast<short> (*tok_iter);
  const stdair::Duration_T iRequestTime (lReqTimeHours, lReqTimeMinutes,
                                         lReqTimeSeconds);

  // Request date-time (aggregation of the two items above)
  const stdair::DateTime_T iRequestDateTime (iRequestDate, iRequestTime);

  // Preferred cabin (e.g., "C")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::CabinCode_T iPreferredCabin (*tok_iter);

  // Party size (e.g., 1)
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::NbOfSeats_T iPartySize = 1;

  // Channel (e.g., "DF")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::ChannelLabel_T iChannel (*tok_iter);

  // Trip type (e.g., "RO")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::TripType_T iTripType (*tok_iter);

  // Stay duration (e.g., 5)
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::DayDuration_T iStayDuration = 5;

  // Frequent flyer (e.g., "NONE")
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::FrequentFlyer_T iFrequentFlyerType ("NONE");

  // Preferred departure time (e.g., "10:00:00")
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lPrefTimeHours = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lPrefTimeMinutes = boost::lexical_cast<short> (*tok_iter);
  ++tok_iter; assert (tok_iter != tokens.end());
  const short lPrefTimeSeconds = boost::lexical_cast<short> (*tok_iter);
  const stdair::Duration_T iPreferredDepartureTime (lPrefTimeHours,
                                                    lPrefTimeMinutes,
                                                    lPrefTimeSeconds);

  // Willingness-to-pay (e.g., 2000.0)
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::WTP_T iWTP = 2000.0;

  // Value of time (e.g., 20.0)
  ++tok_iter; assert (tok_iter != tokens.end());
  const stdair::PriceValue_T iValueOfTime = 20.0;

  // Build and return the booking request structure
  return stdair::BookingRequestStruct (iOrigin,
                                       iDestination, iPOS,
                                       iDepartureDate, iRequestDateTime,
                                       iPreferredCabin, iPartySize,
                                       iChannel, iTripType, iStayDuration,
                                       iFrequentFlyerType,
                                       iPreferredDepartureTime, iWTP,
                                       iValueOfTime);
}

// ///////// M A I N ////////////
int main (int argc, char* argv[]) {

  // State whether the BOM tree should be built-in or parsed from an
  // input file
  bool isBuiltin;

  // A booking request should be given as command-line option
  bool readBookingRequestFromCmdLine;

  // Input file name
  stdair::Filename_T lInputFilename;

  // Output log File
  stdair::Filename_T lLogFilename;

  // Booking request string
  std::string lBookingRequestString;

  // Call the command-line option parser
  const int lOptionParserStatus =
    readConfiguration (argc, argv, isBuiltin,
readBookingRequestFromCmdLine,
                       lInputFilename, lLogFilename, lBookingRequestString);

  if (lOptionParserStatus == K_AIRSCHED_EARLY_RETURN_STATUS
) {
    return 0;
  }

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

  // Initialise the AirSched service object
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  AIRSCHED::AIRSCHED_Service airschedService (
lLogParams);

  // Check wether or not (CSV) input files should be read
  if (isBuiltin == true) {

    // Build the sample BOM tree
    airschedService.buildSampleBom();

  } else {
    // Build the BOM tree from parsing input files
    airschedService.parseAndLoad (lInputFilename);
  }

  // Check wether or not a booking request is given as a command-line option
  if (readBookingRequestFromCmdLine == false) {
    lBookingRequestString = K_AIRSCHED_DEFAULT_BOOKING_REQUEST
;
  }

  // DEBUG
  STDAIR_LOG_DEBUG("Booking request string: '" << lBookingRequestString << "'")
;

  // Create a booking request object
  const stdair::BookingRequestStruct& lBookingRequest =
    parseBookingRequest (lBookingRequestString);

  //
  stdair::TravelSolutionList_T lTravelSolutionList;
  airschedService.buildSegmentPathList (lTravelSolutionList
, lBookingRequest);

  // DEBUG
  STDAIR_LOG_DEBUG ("Parsed booking request: " << lBookingRequest);

  // DEBUG
  std::ostringstream oStream;
  stdair::BomDisplay::csvDisplay (oStream, lTravelSolutionList);
  STDAIR_LOG_DEBUG (oStream.str());

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

  return 0;
}
```

24.15 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

Macros

- #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

24.15.1 Macro Definition Documentation

24.15.1.1 #define BOOST_SPIRIT_DEBUG

Definition at line 12 of file BookingRequestParser.cpp.

24.15.2 Typedef Documentation

24.15.2.1 typedef char char_t

Definition at line 28 of file BookingRequestParser.cpp.

24.15.2.2 typedef char const∗ iterator_t

Definition at line 29 of file BookingRequestParser.cpp.

24.15.2.3 typedef boost::spirit::classic::scanner<iterator_t> scanner_t

Definition at line 31 of file BookingRequestParser.cpp.

24.15.2.4 typedef boost::spirit::classic::rule<scanner_t> rule_t

Definition at line 32 of file BookingRequestParser.cpp.

24.16 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
00037   struct store_place_element {
00039     store_place_element (SearchString_T&
      ioSearchString)
00040       : _searchString (ioSearchString) {}
00041
00043     void operator() (iterator_t iStr, iterator_t
       iStrEnd) const {
00044       std::string lPlace (iStr, iStrEnd);
00045       // std::cout << "Place: " << lPlace << std::endl;
00046
00047       // Set the place
00048       _searchString._tmpPlace._name += " " + lPlace;
00049
00050       // Add the parsed place to the list
00051       // _searchString._placeList.push_back (_searchString._tmpPlace);
00052     }
00053
00054     SearchString_T& _searchString;
00055   };
00056
00058   struct store_date {
00060     store_date (SearchString_T& ioSearchString)
00061       : _searchString (ioSearchString) {}
00062
00064     void operator() (iterator_t iStr, iterator_t
       iStrEnd) const {
00065       _searchString._tmpDate._date = _searchString
      ._tmpDate.getDate();
00066       // std::cout << "Board date: "
00067       // << _searchString._date << std::endl;
```

```

      // Add the parsed date to the list
      _searchString._dateList.push_back (_searchString
._tmpDate);
    }

    SearchString_T& _searchString;
  };

  struct store_airline_sign {
    store_airline_sign (SearchString_T&
ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (bool iAirlineSign) const {
      _searchString._tmpAirline._isPreferred
 = !iAirlineSign;
      // std::cout << "Airline is preferred: " << iAirlineSign << std::endl;
    }

    SearchString_T& _searchString;
  };

  struct store_airline_code {
    store_airline_code (SearchString_T&
ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (iterator_t iStr, iterator_t
 iStrEnd) const {
      std::string lAirlineCode (iStr, iStrEnd);
      _searchString._tmpAirline._code =
lAirlineCode;
      // std::cout << "Airline code: " << lAirlineCode << std::endl;

      // Add the parsed airline to the list
      _searchString._airlineList.push_back (
_searchString._tmpAirline);
    }

    SearchString_T& _searchString;
  };

  struct store_airline_name {
    store_airline_name (SearchString_T&
ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (iterator_t iStr, iterator_t
 iStrEnd) const {
      std::string lAirlineName (iStr, iStrEnd);
      _searchString._tmpAirline._name =
lAirlineName;
      // std::cout << "Airline: " << lAirlineName << std::endl;

      // Add the parsed airline to the list
      _searchString._airlineList.push_back (
_searchString._tmpAirline);
    }

    SearchString_T& _searchString;
  };

  struct store_passenger_number {
    store_passenger_number (SearchString_T&
 ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (unsigned int iNumber) const {
      _searchString._tmpPassenger._number =
iNumber;
      // std::cout << "Number of passengers: " << iNumber << std::endl;
    }

    SearchString_T& _searchString;
  };

  struct store_adult_passenger_type {
    store_adult_passenger_type (SearchString_T
& ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (iterator_t iStr, iterator_t
 iStrEnd) const {
      std::string lPassengerType (iStr, iStrEnd);
      _searchString._tmpPassenger._type =
Passenger_T::ADULT;
```

```cpp
      // std::cout << "Passenger type: " << lPassengerType << std::endl;

      // Add the parsed passenger to the list
      _searchString._passengerList.push_back (_searchString._tmpPassenger);
    }

    SearchString_T& _searchString;
  };

  struct store_child_passenger_type {
    store_child_passenger_type (SearchString_T& ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (iterator_t iStr, iterator_t iStrEnd) const {
      std::string lPassengerType (iStr, iStrEnd);
      _searchString._tmpPassenger._type = Passenger_T::CHILD;
      // std::cout << "Passenger type: " << lPassengerType << std::endl;

      // Add the parsed passenger to the list
      _searchString._passengerList.push_back (_searchString._tmpPassenger);
    }

    SearchString_T& _searchString;
  };

  struct store_pet_passenger_type {
    store_pet_passenger_type (SearchString_T& ioSearchString)
      : _searchString (ioSearchString) {}

    void operator() (iterator_t iStr, iterator_t iStrEnd) const {
      std::string lPassengerType (iStr, iStrEnd);
      _searchString._tmpPassenger._type = Passenger_T::PET;
      // std::cout << "Passenger type: " << lPassengerType << std::endl;

      // Add the parsed passenger to the list
      _searchString._passengerList.push_back (_searchString._tmpPassenger);
    }

    SearchString_T& _searchString;
  };

  // /////////// Utilities /////////////
  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;

  //
  //  Our calculator grammar (using subrules)
  //

  using namespace boost::spirit::classic;

  struct SearchStringParser :
    public boost::spirit::classic::grammar<SearchStringParser> {

    SearchStringParser (SearchString_T& ioSearchString)
      : _searchString (ioSearchString) {
    }

    template <typename ScannerT>
    struct definition {
      definition (SearchStringParser const& self) {

        search_string = places
          >> !( dates )
          >> *( preferred_airlines )
          >> *( passengers )
          ;

        places =
          +( place_element )
          ;

        place_element =
          lexeme_d[ (repeat_p(1,20)[chset_p("a-z")])[store_place_element(self._searchString)] ]
          ;

        dates =
          date[store_date(self._searchString)]
          >> !date[store_date(self._searchString)]
          ;

        date =
          ( month | day )
          >> boost::spirit::classic::chset_p("/-")
          >> ( day | month )
          >> ! ( boost::spirit::classic::chset_p("/-")
                 >> year )
          ;

        day =
          lexeme_d[ limit_d(1u,31u)[uint1_2_p][assign_a(self._searchString._tmpDate._day)] ]
          ;

        month =
          lexeme_d[ limit_d(1u,12u)[uint1_2_p][assign_a(self._searchString._tmpDate._month)] ]
          ;

        year =
          lexeme_d[ limit_d(2000u,2099u)[uint4_p][assign_a(self._searchString._tmpDate._year)] ]
          | lexeme_d[ limit_d(0u,99u)[uint2_p][assign_a(self._searchString._tmpDate._year)] ]
          ;

        preferred_airlines =
          !(boost::spirit::classic::sign_p)[store_airline_sign(self._searchString)]
          >> airline_code | airline_name
          ;

        airline_code =
          lexeme_d[ (repeat_p(2,3)[chset_p("0-9a-z")])[store_airline_code(self._searchString)] ]
          ;

        airline_name =
          lexeme_d[ (repeat_p(4,20)[chset_p("0-9a-z")])[store_airline_name(self._searchString)] ]
          ;

        passengers =
          passenger_number >> passenger_type
          ;

        passenger_number =
          lexeme_d[ limit_d(1u, 9u)[uint1_p][store_passenger_number(self._searchString)] ]
          ;

        passenger_type =
          passenger_adult_type[store_adult_passenger_type(self._searchString)]
          | passenger_child_type[store_child_passenger_type(self._searchString)]
          | passenger_pet_type[store_pet_passenger_type(self._searchString)]
          ;

        passenger_adult_type =
          lexeme_d[ as_lower_d [ str_p("adult") >> !ch_p('s') ] ]
          ;

        passenger_child_type =
          lexeme_d[ as_lower_d [ str_p("child") >> !str_p("ren") ] ]
          ;

        passenger_pet_type =
          lexeme_d[ as_lower_d [ str_p("dog") | str_p("cat") >> !ch_p('s') ] ]
          ;

        BOOST_SPIRIT_DEBUG_NODE (search_string);
        BOOST_SPIRIT_DEBUG_NODE (places);
        BOOST_SPIRIT_DEBUG_NODE (place_element);
        BOOST_SPIRIT_DEBUG_NODE (dates);
        BOOST_SPIRIT_DEBUG_NODE (date);
        BOOST_SPIRIT_DEBUG_NODE (day);
        BOOST_SPIRIT_DEBUG_NODE (month);
        BOOST_SPIRIT_DEBUG_NODE (year);
        BOOST_SPIRIT_DEBUG_NODE (preferred_airlines);
        BOOST_SPIRIT_DEBUG_NODE (airline_code);
        BOOST_SPIRIT_DEBUG_NODE (airline_name);
        BOOST_SPIRIT_DEBUG_NODE (passengers);
        BOOST_SPIRIT_DEBUG_NODE (passenger_number);
        BOOST_SPIRIT_DEBUG_NODE (passenger_type);
        BOOST_SPIRIT_DEBUG_NODE (passenger_adult_type);
        BOOST_SPIRIT_DEBUG_NODE (passenger_child_type);
        BOOST_SPIRIT_DEBUG_NODE (passenger_pet_type);
      }

      boost::spirit::classic::rule<ScannerT> search_string, places,
place_element,
        dates, date, month, day, year,
        preferred_airlines, airline_code, airline_name,
        passengers, passenger_number, passenger_type, passenger_adult_type,
        passenger_child_type, passenger_pet_type;

      boost::spirit::classic::rule<ScannerT> const& start() const { return
 search_string; }
    };

    SearchString_T& _searchString;
  };

  // ///////////////////////////////////////////////////////////
  SearchString_T parseBookingRequest (const
std::string& iSearchString) {
    SearchString_T oSearchStringStruct;

    // Read the search string
    iterator_t lStringIterator = iSearchString.c_str();

    // Instantiate the structure that will hold the result of the parsing.
    SearchStringParser lSearchStringParser (
oSearchStringStruct);
    boost::spirit::classic::parse_info<iterator_t> info =
      boost::spirit::classic::parse (lStringIterator, lSearchStringParser,
                                     boost::spirit::classic::space_p);

    STDAIR_LOG_DEBUG ("-------------------------");

    bool hasBeenParsingSuccessful = info.full;
    if (hasBeenParsingSuccessful == true) {
      STDAIR_LOG_DEBUG ("Parsing succeeded");

    } else {
      STDAIR_LOG_DEBUG ("Parsing failed");
    }
    STDAIR_LOG_DEBUG ("-------------------------");

    return oSearchStringStruct;
  }

}
```

24.17 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)

24.18 BookingRequestParser.hpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>

namespace airsched {

  struct Place_T {
    // Attributes
    std::string _name;
    std::string _code;
    Place_T () : _name (""), _code ("") {}
    /* Display. */
    void display() const {
      std::cout << "Place: " << _name << " (" << _code << ")" <<
std::endl;
    }
  };

  typedef std::vector<Place_T> PlaceList_T;

  struct Date_T {
    // Attributes
    boost::gregorian::date _date;
    unsigned int _reldays;
    unsigned int _day;
    unsigned int _month;
    unsigned int _year;
    Date_T () : _reldays (14), _day(1), _month(1),
_year(1970) {}
    /* Display. */
    void display() const {
      std::cout << "Date: " << _date << " (" << _day << "/" << _month
                << "/" << _year << "), i.e. in " << _reldays << "
 days"
                << std::endl;
    }
    boost::gregorian::date getDate() const {
      return boost::gregorian::date (_year, _month, _day);
    }
  };

  typedef std::vector<Date_T> DateList_T;

  struct Airline_T {
    // Attributes
    bool _isPreferred;
    std::string _name;
    std::string _code;
    Airline_T () : _isPreferred (true), _name(""),
_code("") {}
    /* Display. */
    void display() const {
      const std::string isPreferredStr = (_isPreferred)?"+":"-";
      std::cout << "Airline: " << isPreferredStr << _name << " (" << _code
 << ")"
                << std::endl;
    }
  };

  typedef std::vector<Airline_T> AirlineList_T;

  struct Passenger_T {
    // Attributes
    typedef enum { ADULT = 0, CHILD, PET, LAST_VALUE } PassengerType_T;
    static const std::string _labels[LAST_VALUE];
    PassengerType_T _type;
    unsigned short _number;
    Passenger_T () : _type(ADULT), _number(1) {}
    /* Display. */
    void display() const {
      std::cout << "Passenger: " << _number << " (" << _labels[_type] << ")"
                << std::endl;
    }
  };

  const std::string Passenger_T::_labels[Passenger_T::LAST_VALUE] =
    { "Adult", "Child", "Pet" };

  typedef std::vector<Passenger_T> PassengerList_T;

  struct SearchString_T {
    // Attributes
    PlaceList_T _placeList;
    DateList_T _dateList;
    AirlineList_T _airlineList;
    PassengerList_T _passengerList;

    SearchString_T () {}

    /* Display. */
    void display() const {
      std::cout << std::endl;

      for (PlaceList_T::const_iterator itPlace = _placeList.begin();
           itPlace != _placeList.end(); ++itPlace) {
        const Place_T& lPlace = *itPlace;
        lPlace.display();
      }

      for (DateList_T::const_iterator itDate = _dateList.begin();
           itDate != _dateList.end(); ++itDate) {
        const Date_T& lDate = *itDate;
        lDate.display();
      }

      for (AirlineList_T::const_iterator itAirline = _airlineList.begin();
           itAirline != _airlineList.end(); ++itAirline) {
        const Airline_T& lAirline = *itAirline;
        lAirline.display();
      }

      for (PassengerList_T::const_iterator itPassenger = _passengerList.begin();
           itPassenger != _passengerList.end(); ++itPassenger) {
        const Passenger_T& lPassenger = *itPassenger;
        lPassenger.display();
      }

      std::cout << "-- Staging --" << std::endl;
      _tmpPlace.display();
    }

    // //// Staging ////
    Place_T _tmpPlace;
    Date_T _tmpDate;
    Airline_T _tmpAirline;
    Passenger_T _tmpPassenger;
  };

  //
  //  The booking request grammar (using subrules)
  //

  SearchString_T parseBookingRequest (const std::string& iSearchString);

}
```

24.19 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

24.20 AirportList.hpp

```
#ifndef __AIRSCHED_BOM_AIRPORTLIST_HPP
#define __AIRSCHED_BOM_AIRPORTLIST_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <set>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>

namespace AIRSCHED {

  typedef std::set<stdair::AirportCode_T> AirportList_T;
  typedef std::vector<stdair::AirportCode_T> AirportOrderedList_T
;

}
#endif // __AIRSCHED_BOM_AIRPORTLIST_HPP
```

24.21 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

24.22 BomDisplay.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <ostream>
// StdAir
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
// AirSched
#include <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/BomDisplay.hpp>

namespace AIRSCHED {

  struct FlagSaver {
  public:
    FlagSaver (std::ostream& oStream)
      : _oStream (oStream), _streamFlags (oStream.flags()) {
    }

    ~FlagSaver() {
      // Reset formatting flags of the given output stream
      _oStream.flags (_streamFlags);
    }

  private:
    std::ostream& _oStream;
    std::ios::fmtflags _streamFlags;
  };

  // //////////////////////////////////////////////////////////////////////
  std::string BomDisplay::csvDisplay (const
stdair::BomRoot& iBomRoot) {
    std::ostringstream oStream;

    oStream << std::endl;
    oStream << "============================================================
"
            << std::endl;
    oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
    oStream << "============================================================
"
            << std::endl;

    // Check whether there are ReachableUniverse objects
    if (stdair::BomManager::hasList<ReachableUniverse> (iBomRoot) == false) {
      return oStream.str();
    }

    // Retrieve the ReachableUniverse list
    const ReachableUniverseList_T&
lReachableUniverseList =
      stdair::BomManager::getList<ReachableUniverse> (iBomRoot);

    // Browse the networks for each departure airport
    for (ReachableUniverseList_T::const_iterator itReachableUniverse =
           lReachableUniverseList.begin();
         itReachableUniverse != lReachableUniverseList.end();
         ++itReachableUniverse) {
      ReachableUniverse* lReachableUniverse_ptr = *
itReachableUniverse;
      assert (lReachableUniverse_ptr != NULL);

      // Display the reachable universe
      csvDisplay (oStream, *lReachableUniverse_ptr);
    }

    return oStream.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDisplay (std::ostream& oStream,
                               const ReachableUniverse&
iReachableUniverse) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "+++++++++++++++++++++++++++++++++++++++++++++++" << std::endl
;
    oStream << iReachableUniverse.toString();
    oStream << "+++++++++++++++++++++++++++++++++++++++++++++++" << std::endl
;
  }
}
```

24.23 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

24.24 BomDisplay.hpp

```
#ifndef __AIRSCHED_BOM_BOMDISPLAY_HPP
#define __AIRSCHED_BOM_BOMDISPLAY_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// AirSched

namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  class ReachableUniverse;

  class BomDisplay {
  public:
    // ////////////// Display support methods ////////////////
    static std::string csvDisplay (const stdair::BomRoot&);

    static void csvDisplay (std::ostream&, const ReachableUniverse
&);
  };

}
#endif // __AIRSCHED_BOM_BOMDISPLAY_HPP
```

24.25 airsched/bom/FareFamilyStruct.cpp File Reference

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

Namespaces

- namespace AIRSCHED

24.26 FareFamilyStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// AIRSCHED
#include <airsched/bom/FareFamilyStruct.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  FareFamilyStruct::
  FareFamilyStruct (const stdair::FamilyCode_T& iFamilyCode,
                    const stdair::ClassList_String_T& iClasses)
    : _familyCode (iFamilyCode),
      _classes (iClasses) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string FareFamilyStruct::describe()
 const {
    std::ostringstream ostr;
    ostr << "        " << _familyCode << " " << _classes <<
", ";
    return ostr.str();
  }

}
```

24.27 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

24.28 FareFamilyStruct.hpp

```
#ifndef __AIRSCHED_BOM_FAREFAMILYSTRUCT_HPP
#define __AIRSCHED_BOM_FAREFAMILYSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace AIRSCHED {

  struct FareFamilyStruct : public stdair::StructAbstract {
    // Attributes
    stdair::FamilyCode_T _familyCode;
    stdair::ClassList_String_T _classes;

    FareFamilyStruct (const stdair::FamilyCode_T&,
                      const stdair::ClassList_String_T&);

    const std::string describe() const;
  };

  typedef std::vector<FareFamilyStruct> FareFamilyStructList_T
;

}
#endif // __AIRSCHED_BOM_FAREFAMILYSTRUCT_HPP
```

24.29 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

24.30 FlightPeriodStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  FlightPeriodStruct::FlightPeriodStruct
()
    : _dateRange (stdair::BOOST_DEFAULT_DATE_PERIOD),
      _dow (stdair::DEFAULT_DOW_STRING),
      _legAlreadyDefined (false), _itSeconds (0) {
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::Date_T FlightPeriodStruct::getDate() const
 {
    return stdair::Date_T (_itYear, _itMonth, _itDay);
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::Duration_T FlightPeriodStruct::getTime()
 const {
    return boost::posix_time::hours (_itHours)
      + boost::posix_time::minutes (_itMinutes)
      + boost::posix_time::seconds (_itSeconds);
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string FlightPeriodStruct::describe()
 const {
    std::ostringstream ostr;
    ostr << _airlineCode << _flightNumber << ", " <<
_dateRange
         << " - " << _dow << std::endl;

    for (LegStructList_T::const_iterator itLeg = _legList.begin();
         itLeg != _legList.end(); ++itLeg) {
      const LegStruct& lLeg = *itLeg;
      ostr << lLeg.describe();
    }

    for (SegmentStructList_T::const_iterator itSegment = _segmentList
.begin();
         itSegment != _segmentList.end(); ++itSegment) {
      const SegmentStruct& lSegment = *itSegment;
      ostr << lSegment.describe();
    }

    //ostr << "[Debug] - Staging Leg: ";
    //ostr << _itLeg.describe();
    //ostr << "[Debug] - Staging Cabin: ";
    //ostr << _itCabin.describe();

    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::addAirport (const
stdair::AirportCode_T& iAirport) {
    AirportList_T::const_iterator itAirport = _airportList.find (
iAirport);
    if (itAirport == _airportList.end()) {
      // Add the airport code to the airport set
      const bool insertSuccessful = _airportList.insert (iAirport).
second;

      if (insertSuccessful == false) {
        // TODO: throw an exception
      }

      // Add the airport code to the airport vector
      _airportOrderedList.push_back (iAirport);
    }
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::buildSegments () {
    // The list of airports encompasses all the airports on which
    // the flight takes off or lands. Moreover, that list is
    // time-ordered: the first airport is the initial departure of
    // the flight, and the last airport is the eventual point of
    // rest of the flight.
    // Be l the size of the ordered list of airports.
    // We want to generate all the segment combinations from the legs
    // and, hence, from all the possible (time-ordered) airport pairs.
    // Thus, we both iterator on i=0...l-1 and j=i+1...l
    assert (_airportOrderedList.size() >= 2);

    _segmentList.clear();
    for (AirportOrderedList_T::const_iterator itAirport_i =
           _airportOrderedList.begin();
         itAirport_i != _airportOrderedList.end()-1; ++
itAirport_i) {
      for (AirportOrderedList_T::const_iterator itAirport_j = itAirport_i + 1;
           itAirport_j != _airportOrderedList.end(); ++
itAirport_j) {
        SegmentStruct lSegmentStruct;
        lSegmentStruct._boardingPoint = *itAirport_i;
        lSegmentStruct._offPoint = *itAirport_j;

        _segmentList.push_back (lSegmentStruct);
      }
    }

    // Clear the lists of airports, so that it is ready for the next flight
    _airportList.clear();
    _airportOrderedList.clear();
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::
  addSegmentCabin (const SegmentStruct& iSegment,
                   const SegmentCabinStruct& iCabin) {
    // Retrieve the Segment structure corresponding to the (boarding, off)
 point
    // pair.
    SegmentStructList_T::iterator itSegment = _segmentList.begin();
    for ( ; itSegment != _segmentList.end(); ++itSegment) {
      const SegmentStruct& lSegment = *itSegment;

      const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
;
      const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
      if (lSegment._boardingPoint == lBoardingPoint
          && lSegment._offPoint == lOffPoint) {
        break;
      }
    }

    if (itSegment == _segmentList.end()) {
      std::ostringstream oStr;
      oStr << "Within the schedule input file, there is a flight, for which "
           << "the airports of segments and those of the legs "
           << "do not correspond";
      STDAIR_LOG_ERROR (oStr.str());
      throw SegmentDateNotFoundException (oStr.str(
));
    }

    // Add the Cabin structure to the Segment Cabin structure.
    assert (itSegment != _segmentList.end());
    SegmentStruct& lSegment = *itSegment;
    lSegment._cabinList.push_back (iCabin);
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::
  addSegmentCabin (const SegmentCabinStruct&
 iCabin) {
    // Iterate on all the Segment structures (as they get the same cabin
    // definitions)
    for (SegmentStructList_T::iterator itSegment = _segmentList.
begin();
         itSegment != _segmentList.end(); ++itSegment) {
      SegmentStruct& lSegment = *itSegment;

      lSegment._cabinList.push_back (iCabin);
    }
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::
  addFareFamily (const SegmentStruct& iSegment,
                 const SegmentCabinStruct& iCabin,
                 const FareFamilyStruct& iFareFamily) {
    // Retrieve the Segment structure corresponding to the (boarding, off)
 point
    // pair.
    SegmentStructList_T::iterator itSegment = _segmentList.begin();
    for ( ; itSegment != _segmentList.end(); ++itSegment) {
      const SegmentStruct& lSegment = *itSegment;

      const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
;
      const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
      if (lSegment._boardingPoint == lBoardingPoint
          && lSegment._offPoint == lOffPoint) {
        break;
      }
    }

    if (itSegment == _segmentList.end()) {
      std::ostringstream oStr;
      oStr << "Within the schedule input file, there is a flight, for which "
           << "the airports of segments and those of the legs "
           << "do not correspond";
      STDAIR_LOG_ERROR (oStr.str());
      throw SegmentDateNotFoundException (oStr.str(
));
    }

    // Add the Cabin structure to the Segment Cabin structure.
    assert (itSegment != _segmentList.end());
    SegmentStruct& lSegment = *itSegment;

    // Retrieve the Segment cabin structure given the cabin code
    SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.
begin();
    for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
      const SegmentCabinStruct& lCabin = *itCabin;

      const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
      if (iCabin._cabinCode == lCabinCode) {
        break;
      }
    }

    if (itCabin == lSegment._cabinList.end()) {
      std::ostringstream oStr;
      oStr << "Within the schedule input file, there is a flight "
           << "for which the cabin code does not exist.";
      STDAIR_LOG_ERROR (oStr.str());
      throw SegmentDateNotFoundException (oStr.str());
    }

    // Add the Cabin structure to the Segment Cabin structure.
    assert (itCabin != lSegment._cabinList.end());
    SegmentCabinStruct& lCabin = *itCabin;
    lCabin._fareFamilies.push_back(iFareFamily);
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodStruct::
  addFareFamily (const SegmentCabinStruct& iCabin,
                 const FareFamilyStruct& iFareFamily) {
    // Iterate on all the Segment structures (as they get the same cabin
    // definitions)

    for (SegmentStructList_T::iterator itSegment = _segmentList.begin();
         itSegment != _segmentList.end(); ++itSegment) {
      SegmentStruct& lSegment = *itSegment;

      // Retrieve the Segment cabin structure given the cabin code
      SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.begin();
      for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
        const SegmentCabinStruct& lCabin = *itCabin;

        const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
        if (iCabin._cabinCode == lCabinCode) {
          break;
        }
      }

      if (itCabin == lSegment._cabinList.end()) {
        std::ostringstream oStr;
        oStr << "Within the schedule input file, there is a flight "
             << "for which the cabin code does not exist.";
        STDAIR_LOG_ERROR (oStr.str());
        throw SegmentDateNotFoundException (oStr.str());
      }

      // Add the Cabin structure to the Segment Cabin structure.
      assert (itCabin != lSegment._cabinList.end());
      SegmentCabinStruct& lCabin = *itCabin;
      lCabin._fareFamilies.push_back(iFareFamily);
    }
  }

}
```

24.31 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

24.32 FlightPeriodStruct.hpp

```
#ifndef __AIRSCHED_BOM_FLIGHTPERIODSTRUCT_HPP
#define __AIRSCHED_BOM_FLIGHTPERIODSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
// AirSched
#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>

namespace AIRSCHED {

  struct FlightPeriodStruct : public stdair::StructAbstract {

    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();

    // 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;
  };

}
#endif // __AIRSCHED_BOM_FLIGHTPERIODSTRUCT_HPP
```

24.33 airsched/bom/LegCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/LegCabin.hpp>
#include <airsched/bom/LegCabinStruct.hpp>
```

Namespaces

- namespace AIRSCHED

24.34 LegCabinStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// STDAIR
#include <stdair/bom/LegCabin.hpp>
// AIRSCHED
#include <airsched/bom/LegCabinStruct.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  const std::string LegCabinStruct::describe() const {
    std::ostringstream ostr;
    ostr << "        " << _cabinCode << " " << _capacity <<
", ";
    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void LegCabinStruct::fill (stdair::LegCabin& ioLegCabin)
 const {
    // Set the Capacity
    ioLegCabin.setCapacities (_capacity);
  }

}
```

24.35 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

24.36 LegCabinStruct.hpp

```
#ifndef __AIRSCHED_BOM_LEGCABINSTRUCT_HPP
#define __AIRSCHED_BOM_LEGCABINSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

// Forward declarations
namespace stdair {
  class LegCabin;
}

namespace AIRSCHED {

  struct LegCabinStruct : public stdair::StructAbstract {
    // Attributes
    stdair::CabinCode_T _cabinCode;
    stdair::CabinCapacity_T _capacity;

    void fill (stdair::LegCabin&) const;

    const std::string describe() const;
  };

  typedef std::vector<LegCabinStruct> LegCabinStructList_T;

}
#endif // __AIRSCHED_BOM_LEGCABINSTRUCT_HPP
```

24.37 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

24.38 LegStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// STDAIR
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/LegDate.hpp>
// AIRSCHED
#include <airsched/bom/LegStruct.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  LegStruct::LegStruct ()
    : _boardingDateOffset (stdair::DEFAULT_DATE_OFFSET),
      _offDateOffset (stdair::DEFAULT_DATE_OFFSET) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string LegStruct::describe() const {
    std::ostringstream ostr;
    ostr << "    " << _boardingPoint << " / "
         << boost::posix_time::to_simple_string(_boardingTime);
    if (_boardingDateOffset.days() != 0) {
      ostr << " [" << _boardingDateOffset.days() << "]";
    }
    ostr << " -- " << _offPoint << " / "
         << boost::posix_time::to_simple_string(_offTime);
    if (_offDateOffset.days() != 0) {
      ostr << " [" << _offDateOffset.days() << "]";
    }
    ostr << " --> "
         << boost::posix_time::to_simple_string(_elapsed)
         << std::endl;
    for (LegCabinStructList_T::const_iterator itCabin = _cabinList.
begin();
         itCabin != _cabinList.end(); itCabin++) {
      const LegCabinStruct& lCabin = *itCabin;
      ostr << lCabin.describe();
    }
    ostr << std::endl;

    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void LegStruct::fill (const stdair::Date_T& iRefDate,
                        stdair::LegDate& ioLegDate) const {
    // Set the Off Point
    ioLegDate.setOffPoint (_offPoint);

    // Set the Boarding Date
    ioLegDate.setBoardingDate (iRefDate + _boardingDateOffset
);

    // Set the Boarding Time
    ioLegDate.setBoardingTime (_boardingTime);

    // Set the Off Date
    ioLegDate.setOffDate (iRefDate + _offDateOffset);

    // Set the Off Time
    ioLegDate.setOffTime (_offTime);

    // Set the Elapsed Time
    ioLegDate.setElapsedTime (_elapsed);
  }

}
```

24.39 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

24.40 LegStruct.hpp

```
#ifndef __AIRSCHED_BOM_LEGSTRUCT_HPP
#define __AIRSCHED_BOM_LEGSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
// AirSched
#include <airsched/bom/LegCabinStruct.hpp>

// Forward declarations
namespace stdair {
  class LegDate;
}

namespace AIRSCHED {

  struct LegStruct : public stdair::StructAbstract {
    // 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;

    void fill (const stdair::Date_T& iRefDate, stdair::LegDate&) const;

    const std::string describe() const;

    LegStruct();
  };

  typedef std::vector<LegStruct> LegStructList_T;

}
#endif // __AIRSCHED_BOM_LEGSTRUCT_HPP
```

24.41 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

24.42 OnDPeriodStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <iostream>
// StdAir
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
// AIRSCHED
#include <airsched/bom/OnDPeriodStruct.hpp>

namespace AIRSCHED {
  // //////////////////////////////////////////////////////////////////////
  OnDPeriodStruct::OnDPeriodStruct ()
    : _datePeriod (stdair::BOOST_DEFAULT_DATE_PERIOD),
      _timeRangeStart (stdair::NULL_BOOST_TIME_DURATION),
      _timeRangeEnd (stdair::NULL_BOOST_TIME_DURATION),
      _nbOfAirlines(stdair::DEFAULT_NBOFAIRLINES),
      _airlineCode (stdair::DEFAULT_NULL_AIRLINE_CODE),
      _classCode (stdair::DEFAULT_NULL_CLASS_CODE),
      _itSeconds (0) {
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::Date_T OnDPeriodStruct::getDate() const {
    return stdair::Date_T (_itYear, _itMonth, _itDay);
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::Duration_T OnDPeriodStruct::getTime() const {
    return boost::posix_time::hours (_itHours)
      + boost::posix_time::minutes (_itMinutes)
      + boost::posix_time::seconds (_itSeconds);
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string OnDPeriodStruct::describe() const
{
    std::ostringstream ostr;
    ostr << _origin << "-" << _destination << ", "
         << _datePeriod << ", between  "
         << boost::posix_time::to_simple_string(_timeRangeStart)
         << " to "
         << boost::posix_time::to_simple_string(_timeRangeEnd) <<
", "
         << _classCode << ", "
         << _airlineCode << ", "
         << std::endl;

    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string OnDPeriodStruct::describeTSKey
() const {
    std::ostringstream ostr;
    ostr << _origin << "-" << _destination << ", "
```

```
        << _airlineCode << ", " << _classCode <<
std::endl;

    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const stdair::AirlineCode_T& OnDPeriodStruct::getFirstAirlineCode
 () const {
    assert (_airlineCodeList.size() > 0);
    stdair::AirlineCodeList_T::const_iterator itFirstAirlineCode =
      _airlineCodeList.begin();
    return *itFirstAirlineCode;
  }

}
```

24.43 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

24.44 OnDPeriodStruct.hpp

```
#ifndef __AIRSCHED_BOM_ONDPERIODSTRUCT_HPP
#define __AIRSCHED_BOM_ONDPERIODSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace AIRSCHED {
  struct OnDPeriodStruct : public stdair::StructAbstract {
  public:
    // ////////// Getters /////////////
    const stdair::AirlineCode_T& getFirstAirlineCode ()
const;

    stdair::Date_T getDate() const;

    stdair::Duration_T getTime() const;

    // ///////// Display Methods //////////
    const std::string describe() const;

    const std::string describeTSKey() const;

  public:
    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;
  };
}
#endif // __AIRSCHED_BOM_ONDPERIODSTRUCT_HPP
```

24.45 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)

24.46 OriginDestinationSet.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
// AirSched
#include <airsched/bom/OriginDestinationSet.hpp
>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  OriginDestinationSet::OriginDestinationSet()
    : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  OriginDestinationSet::OriginDestinationSet (const OriginDestinationSet&)
    : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSet::OriginDestinationSet (const Key_T& iKey)
    : _key (iKey), _parent (NULL) {
  }

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSet::~OriginDestinationSet
() {
  }

  // //////////////////////////////////////////////////////////////////////
  std::string OriginDestinationSet::toString()
 const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSet::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSet::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // //////////////////////////////////////////////////////////////////////
  template<class Archive>
  void OriginDestinationSet::serialize (Archive&
 ioArchive,
                                        const unsigned int iFileVersion) {
    ioArchive & _key;
  }

  // //////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template
  void OriginDestinationSet::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                            unsigned int);
  template
  void OriginDestinationSet::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                            unsigned int);
  // //////////////////////////////////////////////////////////////////////

}

```

24.47 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

24.48 OriginDestinationSet.hpp

```
#ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSET_HPP
#define __AIRSCHED_BOM_ORIGINDESTINATIONSET_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/BomAbstract.hpp>
// AirSched
#include <airsched/bom/OriginDestinationSetKey.hpp
>
#include <airsched/bom/OriginDestinationSetTypes.hpp
>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {
  template <typename BOM> class FacBom;
  class FacBomManager;
}

namespace AIRSCHED {

  class OriginDestinationSet : public stdair::BomAbstract {
    template <typename BOM> friend class stdair::FacBom;
    friend  class stdair::FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef OriginDestinationSetKey Key_T;


  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    const stdair::AirportCode_T& getDestination() const {
      return _key.getOffPoint();
    }

    stdair::BomAbstract* const getParent() const {
      return _parent;
    }

    const stdair::HolderMap_T& getHolderMap() const {
      return _holderMap;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  protected:
    // ////////// Constructors and destructors /////////
    OriginDestinationSet (const Key_T&);

    ~OriginDestinationSet();

  private:
    OriginDestinationSet();

    OriginDestinationSet (const OriginDestinationSet
&);

  protected:
    // ////////// Attributes /////////
    Key_T _key;

    stdair::BomAbstract* _parent;

    stdair::HolderMap_T _holderMap;
  };

}
#endif // __AIRSCHED_BOM_ORIGINDESTINATIONSET_HPP

```

24.49 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)

24.50 OriginDestinationSetKey.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
```

```
#include <stdair/basic/BasConst_Inventory.hpp>
// AirSched
#include <airsched/bom/OriginDestinationSetKey.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSetKey::OriginDestinationSetKey()
    : _destination (stdair::DEFAULT_DESTINATION) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSetKey::
  OriginDestinationSetKey (const stdair::AirportCode_T& iDestination)
    : _destination (iDestination) {
  }

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSetKey::
  OriginDestinationSetKey (const OriginDestinationSetKey
& iKey)
    : _destination (iKey._destination) {
  }

  // //////////////////////////////////////////////////////////////////////
  OriginDestinationSetKey::~OriginDestinationSetKey
() {
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSetKey::toStream (
std::ostream& ioOut) const {
    ioOut << "OriginDestinationSetKey: " << toString() << std::endl;
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSetKey::fromStream (
std::istream& ioIn) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string OriginDestinationSetKey::toString
() const {
    std::ostringstream oStr;
    oStr << _destination;
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSetKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // //////////////////////////////////////////////////////////////////////
  void OriginDestinationSetKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // //////////////////////////////////////////////////////////////////////
  template<class Archive>
  void OriginDestinationSetKey::serialize (
Archive& ioArchive,
                                            const unsigned int iFileVersion) {
    ioArchive & _destination;
  }

  // //////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template
  void OriginDestinationSetKey::serialize<ba::text_oarchive> (ba::text_oarchive
&,
                                                               unsigned int);
  template
  void OriginDestinationSetKey::serialize<ba::text_iarchive> (ba::text_iarchive
&,
                                                               unsigned int);
  // //////////////////////////////////////////////////////////////////////

}
```

24.51 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

24.52 OriginDestinationSetKey.hpp

```
#ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSETKEY_HPP
#define __AIRSCHED_BOM_ORIGINDESTINATIONSETKEY_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace AIRSCHED {

  struct OriginDestinationSetKey : public
stdair::KeyAbstract {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  private:
    OriginDestinationSetKey();

  public:
    OriginDestinationSetKey (const stdair::AirportCode_T
& iDestination);

    OriginDestinationSetKey (const
OriginDestinationSetKey&);

    ~OriginDestinationSetKey();


  public:
    // /////////// Getters //////////
    const stdair::AirportCode_T& getOffPoint() const {
      return _destination;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  private:
    // ///////////////// Attributes ///////////////
    stdair::AirportCode_T _destination;
  };

}
#endif // __AIRSCHED_BOM_ORIGINDESTINATIONSETKEY_HPP
```

24.53 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

24.54 OriginDestinationSetTypes.hpp

```
// //////////////////////////////////////////////////////////////////////
#ifndef __AIRSCHED_BOM_ORIGINDESTINATIONSETTYPES_HPP
#define __AIRSCHED_BOM_ORIGINDESTINATIONSETTYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>

namespace AIRSCHED {

  // Forward declarations.
  class OriginDestinationSet;

  typedef std::list<OriginDestinationSet*> OriginDestinationSetList_T
;

  typedef std::map<const stdair::MapKey_T,
                   OriginDestinationSet*>
OriginDestinationSetMap_T;

}
#endif // __AIRSCHED_BOM_ORIGINDESTINATIONSETTYPES_HPP

```

24.55 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)

24.56 ReachableUniverse.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
// AirSched
#include <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/SegmentPathPeriod.hpp>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  ReachableUniverse::ReachableUniverse()
    : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  ReachableUniverse::ReachableUniverse (const ReachableUniverse&)
    : _key (stdair::DEFAULT_ORIGIN), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  ReachableUniverse::ReachableUniverse (const Key_T& iKey)
    : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  ReachableUniverse::~ReachableUniverse()
{
  }

  // //////////////////////////////////////////////////////////////////////
  std::string ReachableUniverse::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverse::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverse::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // //////////////////////////////////////////////////////////////////////
  template<class Archive>
  void ReachableUniverse::serialize (Archive&
ioArchive,
                                     const unsigned int iFileVersion) {
    ioArchive & _key;
  }

  // //////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template
  void ReachableUniverse::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                        unsigned int);
  template
  void ReachableUniverse::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                        unsigned int);
  // //////////////////////////////////////////////////////////////////////

}

```

24.57 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

24.58 ReachableUniverse.hpp

```
#ifndef __AIRSCHED_BOM_REACHABLEUNIVERSE_HPP
#define __AIRSCHED_BOM_REACHABLEUNIVERSE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/BomAbstract.hpp>
// AirSched
#include <airsched/bom/ReachableUniverseKey.hpp
>
#include <airsched/bom/ReachableUniverseTypes.hpp
>
#include <airsched/bom/SegmentPathPeriodTypes.hpp
>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {
  template <typename BOM> class FacBom;
  class FacBomManager;
}

namespace AIRSCHED {

  class ReachableUniverse : public stdair::BomAbstract {
    template <typename BOM> friend class stdair::FacBom;
    friend class stdair::FacBomManager;
    friend class SegmentPathGenerator;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef ReachableUniverseKey Key_T;

  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    const stdair::AirportCode_T& getOrigin() const {
      return _key.getBoardingPoint();
    }

    stdair::BomAbstract* const getParent() const {
      return _parent;
    }

    const stdair::HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const SegmentPathPeriodListList_T&
getSegmentPathPeriodListList() const {
      return _segmentPathPeriodListList;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  protected:
    // ////////// Constructors and destructors /////////
    ReachableUniverse (const Key_T&);

    ~ReachableUniverse();

  private:
    ReachableUniverse();

    ReachableUniverse (const ReachableUniverse
&);


  protected:
    // ////////// Attributes /////////
    Key_T _key;

    stdair::BomAbstract* _parent;

    stdair::HolderMap_T _holderMap;

    SegmentPathPeriodListList_T
_segmentPathPeriodListList;
  };

}
#endif // __AIRSCHED_BOM_REACHABLEUNIVERSE_HPP

```

24.59 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)

24.60 ReachableUniverseKey.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
```

```
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
// AirSched
#include <airsched/bom/ReachableUniverseKey.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  ReachableUniverseKey::ReachableUniverseKey()
    : _origin (stdair::DEFAULT_ORIGIN) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  ReachableUniverseKey::
  ReachableUniverseKey (const ReachableUniverseKey& iKey)
    : _origin (iKey._origin) {
  }

  // //////////////////////////////////////////////////////////////////////
  ReachableUniverseKey::
  ReachableUniverseKey (const stdair::AirportCode_T& iAirportCode)
    : _origin (iAirportCode) {
  }

  // //////////////////////////////////////////////////////////////////////
  ReachableUniverseKey::~ReachableUniverseKey
() {
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverseKey::toStream (
std::ostream& ioOut) const {
    ioOut << "ReachableUniverseKey: " << toString() << std::endl;
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverseKey::fromStream (
std::istream& ioIn) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string ReachableUniverseKey::toString
() const {
    std::ostringstream oStr;
    oStr << _origin;
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverseKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // //////////////////////////////////////////////////////////////////////
  void ReachableUniverseKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // //////////////////////////////////////////////////////////////////////
  template<class Archive>
  void ReachableUniverseKey::serialize (Archive&
 ioArchive,
                                        const unsigned int iFileVersion) {
    ioArchive & _origin;
  }

  // //////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template
  void ReachableUniverseKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                            unsigned int);
  template
  void ReachableUniverseKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                            unsigned int);
  // //////////////////////////////////////////////////////////////////////

}
```

24.61 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

24.62 ReachableUniverseKey.hpp

```
#ifndef __AIRSCHED_BOM_REACHABLEUNIVERSEKEY_HPP
#define __AIRSCHED_BOM_REACHABLEUNIVERSEKEY_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace AIRSCHED {

  struct ReachableUniverseKey : public stdair::KeyAbstract
 {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  private:
    ReachableUniverseKey();

  public:
    ReachableUniverseKey (const stdair::AirportCode_T&
iOrigin);

    ReachableUniverseKey (const ReachableUniverseKey
&);

    ~ReachableUniverseKey();


  public:
    // /////////// Getters //////////
    const stdair::AirportCode_T& getBoardingPoint() const {
      return _origin;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  private:
    // ///////////////// Attributes ///////////////
    stdair::AirportCode_T _origin;
  };

}

#endif // __AIRSCHED_BOM_REACHABLEUNIVERSEKEY_HPP
```

24.63 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

24.64 ReachableUniverseTypes.hpp

```
// //////////////////////////////////////////////////////////////////////
#ifndef __AIRSCHED_BOM_REACHABLEUNIVERSETYPES_HPP
#define __AIRSCHED_BOM_REACHABLEUNIVERSETYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>

namespace AIRSCHED {

  // Forward declarations.
  class ReachableUniverse;

  typedef std::list<ReachableUniverse*> ReachableUniverseList_T
;

  typedef std::map<const stdair::MapKey_T,
                   ReachableUniverse*> ReachableUniverseMap_T
;
}
#endif // __AIRSCHED_BOM_REACHABLEUNIVERSETYPES_HPP
```

24.65 airsched/bom/SegmentCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentCabin.hpp>
#include <airsched/bom/SegmentCabinStruct.hpp>
```

Namespaces

- namespace AIRSCHED

24.66 SegmentCabinStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// STDAIR
#include <stdair/bom/SegmentCabin.hpp>
// AIRSCHED
#include <airsched/bom/SegmentCabinStruct.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  const std::string SegmentCabinStruct::describe()
 const {
    std::ostringstream ostr;
    ostr << "         " << _cabinCode << " " << _classes;
    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentCabinStruct::fill (stdair::SegmentCabin&
ioSegmentCabin) const {
  }

}
```

24.67 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

24.68 SegmentCabinStruct.hpp

```
#ifndef __AIRSCHED_BOM_SEGMENTCABINSTRUCT_HPP
#define __AIRSCHED_BOM_SEGMENTCABINSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
// AirSched
#include <airsched/bom/FareFamilyStruct.hpp>

// Forward declarations
namespace stdair {
  class SegmentCabin;
}

namespace AIRSCHED {

  struct SegmentCabinStruct : public stdair::StructAbstract {
    // Attributes
    stdair::CabinCode_T _cabinCode;
    stdair::ClassList_String_T _classes;
    stdair::FamilyCode_T _itFamilyCode;
    FareFamilyStructList_T _fareFamilies;

    void fill (stdair::SegmentCabin&) const;

    const std::string describe() const;

  };

  typedef std::vector<SegmentCabinStruct> SegmentCabinStructList_T
;

}
#endif // __AIRSCHED_BOM_SEGMENTCABINSTRUCT_HPP
```

24.69 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

Functions

- template void AIRSCHED::SegmentPathPeriod::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void AIRSCHED::SegmentPathPeriod::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

24.70 SegmentPathPeriod.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#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>
// AirSched
#include <airsched/bom/SegmentPathPeriod.hpp>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  SegmentPathPeriod::SegmentPathPeriod()
    :  _key (stdair::PeriodStruct (stdair::BOOST_DEFAULT_DATE_PERIOD,
                                   stdair::DEFAULT_DOW_STRING),
             stdair::NULL_BOOST_TIME_DURATION, stdair::NULL_BOOST_TIME_DURATION
,
             DateOffsetList_T(),
             stdair::DEFAULT_NBOFAIRLINES),
       _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPathPeriod::SegmentPathPeriod (const SegmentPathPeriod& iSPP)
    :  _key (iSPP._key), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPathPeriod::SegmentPathPeriod (const Key_T& iKey)
    :  _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPathPeriod::~SegmentPathPeriod()
{
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SegmentPathPeriod::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathPeriod::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

```

```
// //////////////////////////////////////////////////////////////////////
void SegmentPathPeriod::serialisationImplementationImport() {
  std::istringstream iStr;
  boost::archive::text_iarchive ia (iStr);
  ia >> *this;
}

// //////////////////////////////////////////////////////////////////////
template<class Archive>
void SegmentPathPeriod::serialize (Archive& ioArchive,
                                   const unsigned int iFileVersion) {
  ioArchive & _key;
}

// //////////////////////////////////////////////////////////////////////
// Explicit template instantiation
namespace ba = boost::archive;
template
void SegmentPathPeriod::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                      unsigned int);
template
void SegmentPathPeriod::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                      unsigned int);
// //////////////////////////////////////////////////////////////////////

// //////////////////////////////////////////////////////////////////////
stdair::SegmentPeriod* SegmentPathPeriod::getLastSegmentPeriod () const {
  // Retrieve the last segment of the list
  const stdair::SegmentPeriodList_T& lSegmentPeriodList =
    stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
  stdair::SegmentPeriodList_T::const_reverse_iterator itLastSegment =
    lSegmentPeriodList.rbegin();

  if (itLastSegment == lSegmentPeriodList.rend()) {
    return NULL;
  }

  stdair::SegmentPeriod* oSegment_ptr = *itLastSegment;
  assert (oSegment_ptr != NULL);

  return oSegment_ptr;
}

// //////////////////////////////////////////////////////////////////////
stdair::SegmentPeriod* SegmentPathPeriod::getFirstSegmentPeriod () const{
  // Retrieve the first segment of the list
  const stdair::SegmentPeriodList_T& lSegmentPeriodList =
    stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
  stdair::SegmentPeriodList_T::const_iterator itFirstSegment =
    lSegmentPeriodList.begin();

  if (itFirstSegment == lSegmentPeriodList.end()) {
    return NULL;
  }

  stdair::SegmentPeriod* oSegment_ptr = *itFirstSegment;
  assert (oSegment_ptr != NULL);

  return oSegment_ptr;
}

// //////////////////////////////////////////////////////////////////////
const stdair::AirportCode_T& SegmentPathPeriod::getDestination () const {
  const stdair::SegmentPeriod* lLastSegment_ptr = getLastSegmentPeriod();
  assert (lLastSegment_ptr != NULL);
  return lLastSegment_ptr->getOffPoint();
}

// //////////////////////////////////////////////////////////////////////
bool SegmentPathPeriod::
isAirlineFlown (const stdair::AirlineCode_T& iAirlineCode) const {
  bool oAirlineFlown = false;

  const stdair::SegmentPeriodList_T& lSegmentPeriodList =
    stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
  for (stdair::SegmentPeriodList_T::const_iterator itSegmentPeriod =
         lSegmentPeriodList.begin();
       itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
    const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
    assert (lSegmentPeriod_ptr != NULL);

```

```
      const stdair::FlightPeriod& lFlightPeriod =
        stdair::BomManager::getParent<stdair::FlightPeriod>(*lSegmentPeriod_ptr);
      const stdair::Inventory& lInventory =
        stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
      const stdair::AirlineCode_T& lSegmentAirlineCode =
        lInventory.getAirlineCode ();
      if (lSegmentAirlineCode == iAirlineCode) {
        oAirlineFlown = true;
        break;
      }
    }

    return oAirlineFlown;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPathPeriodKey SegmentPathPeriod::
  connectWithAnotherSegment(const SegmentPathPeriod& iSingleSegmentPath) const {
    SegmentPathPeriodKey oSegmentPathPeriodKey;

    // Retrieve the (only) segment period of the single segment path.
    const stdair::SegmentPeriod* lNextSegmentPeriod_ptr =
      iSingleSegmentPath.getFirstSegmentPeriod();
    assert (lNextSegmentPeriod_ptr != NULL);

    // Retrive the last segment period of the current segment path and check
    // if the combination of the last segment and the next segment that we
    // want to add to the current segment path will create a new segment
    // (i.e., the two segment period belongs to the same flight number).
    const stdair::SegmentPeriod* lLastSegmentPeriod_ptr = getLastSegmentPeriod ();
    assert (lLastSegmentPeriod_ptr != NULL);
    const stdair::FlightPeriod& lLastFlightPeriod = stdair::BomManager::
      getParent<stdair::FlightPeriod> (*lLastSegmentPeriod_ptr);
    const stdair::Inventory& lLastInventory =
      stdair::BomManager::getParent<stdair::Inventory> (lLastFlightPeriod);

    const stdair::FlightPeriod& lNextFlightPeriod = stdair::BomManager::
      getParent<stdair::FlightPeriod> (*lNextSegmentPeriod_ptr);
    const stdair::Inventory& lNextInventory =
      stdair::BomManager::getParent<stdair::Inventory> (lNextFlightPeriod);

    if (lLastFlightPeriod.getFlightNumber()==lNextFlightPeriod.getFlightNumber()
        && lLastInventory.getAirlineCode() == lNextInventory.getAirlineCode()) {
      return oSegmentPathPeriodKey;
    }

    // Check if the new segment period will create a circle.
    const stdair::AirportCode_T& lDestination =
      lNextSegmentPeriod_ptr->getOffPoint();
    if (checkCircle (lDestination) == true) {
      return oSegmentPathPeriodKey;
    }

    // Check if a passenger can connect from the last segment of the
    // current segment path to the first segment of the to-be-added
    // segment path. If yes, build a new departure period for the new
    // segment path.
    DateOffsetList_T lBoardingDateOffsetList =
      getBoardingDateOffsetList();
    const stdair::PeriodStruct& lCurrentDeparturePeriod = getDeparturePeriod();
    const stdair::PeriodStruct& lNextDeparturePeriod =
      iSingleSegmentPath.getDeparturePeriod();
    const stdair::Duration_T& lLastOffTime =
      lLastSegmentPeriod_ptr->getOffTime();
    const stdair::Duration_T& lNextBoardingTime =
      lNextSegmentPeriod_ptr->getBoardingTime();
    // If the next boarding time is later than the last off time, check if
    // the passengers will have enough time for the transfer. If the next
    // boarding time is earlier than the last off time, check if the passengers
    // can connect to a flight in the next day.
    if (lNextBoardingTime >= lLastOffTime) {
      const stdair::Duration_T lStopTime = lNextBoardingTime - lLastOffTime;
      if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
        return oSegmentPathPeriodKey;
      } else {
        // Calulcate the date offset of the next segment compare to
        // the first one. In this case, this value is equal to the offset
        // of the off date of the last segment compare to the boarding date
        // of the first segment.
        const stdair::DateOffset_T& lLastBoardingDateOffset =
          lBoardingDateOffsetList.at (getNbOfSegments() - 1);
        const stdair::DateOffset_T lNextBoardingDateOffset =
          lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
          - lLastSegmentPeriod_ptr->getBoardingDateOffset();
        const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
          stdair::DateOffset_T (0) - lNextBoardingDateOffset;

        // Compute the adjusted departure period of the next segment by
        // substracting the origin one with the boarding date offset.
        const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
          lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);

        // Build the intersection of the two periods.
        const stdair::PeriodStruct lNewDeparturePeriod =
          lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
        stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
          lNextSegmentPeriod_ptr->getElapsedTime();
        lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
        oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
        oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
      }
    } else {
      const stdair::Duration_T lStopTime =
        lNextBoardingTime - lLastOffTime + stdair::Duration_T (24, 0, 0);
      if (lStopTime < stdair::DEFAULT_MINIMAL_CONNECTION_TIME) {
        return oSegmentPathPeriodKey;
      } else {
        // Calulcate the date offset of the next segment compare to
        // the first one.
        const stdair::DateOffset_T& lLastBoardingDateOffset =
          lBoardingDateOffsetList.at (getNbOfSegments() - 1);
        const stdair::DateOffset_T lNextBoardingDateOffset =
          lLastBoardingDateOffset + lLastSegmentPeriod_ptr->getOffDateOffset()
          - lLastSegmentPeriod_ptr->getBoardingDateOffset() +
          stdair::DateOffset_T (1);
        const stdair::DateOffset_T lNegativeNextBoardingDateOffset =
          stdair::DateOffset_T (0) - lNextBoardingDateOffset;

        // Compute the adjusted departure period of the next segment by
        // substracting the origin one with the boarding date offset.
        const stdair::PeriodStruct lAdjustedNextDeparturePeriod =
          lNextDeparturePeriod.addDateOffset (lNegativeNextBoardingDateOffset);

        // Build the intersection of the two periods.
        const stdair::PeriodStruct lNewDeparturePeriod =
          lCurrentDeparturePeriod.intersection (lAdjustedNextDeparturePeriod);
        stdair::Duration_T lNewElapsed = getElapsedTime() + lStopTime +
          lNextSegmentPeriod_ptr->getElapsedTime();
        lBoardingDateOffsetList.push_back (lNextBoardingDateOffset);
        oSegmentPathPeriodKey.setPeriod (lNewDeparturePeriod);
        oSegmentPathPeriodKey.setElapsedTime (lNewElapsed);
      }
    }

    const stdair::Duration_T& lBoardingTime = getBoardingTime();
    oSegmentPathPeriodKey.setBoardingTime (lBoardingTime);
    oSegmentPathPeriodKey.setBoardingDateOffsetList (lBoardingDateOffsetList);

    return oSegmentPathPeriodKey;
  }

  // ////////////////////////////////////////////////////////////////////
  bool SegmentPathPeriod::
  checkCircle (const stdair::AirlineCode_T& iDestination) const {
    const stdair::SegmentPeriodList_T& lSegmentPeriodList =
      stdair::BomManager::getList<stdair::SegmentPeriod> (*this);
    for (stdair::SegmentPeriodList_T::const_iterator itSegment =
           lSegmentPeriodList.begin();
         itSegment != lSegmentPeriodList.end(); ++itSegment) {
      const stdair::SegmentPeriod* lCurrentSegment_ptr = *itSegment;
      assert (lCurrentSegment_ptr != NULL);
      const stdair::AirlineCode_T& lCurrentBoardingPoint =
        lCurrentSegment_ptr->getBoardingPoint();
      if (lCurrentBoardingPoint == iDestination) {
        return true;
      }
    }
    return false;
  }

  // ////////////////////////////////////////////////////////////////////
  bool SegmentPathPeriod::
  isDepartureDateValid (const stdair::Date_T& iDepartureDate) const {
    const stdair::PeriodStruct& lPeriod = getDeparturePeriod
    ();

    // Check if the departure date is within the date range.
    const stdair::DatePeriod_T& lDeparturePeriod = lPeriod.getDateRange ();
    if (lDeparturePeriod.contains (iDepartureDate) == false) {
      return false;
    }

    // Check if the departure date is valid within the DOW.
    // 0 = Sunday, 1 = Monday, etc.
    const short lDay = iDepartureDate.day_of_week ();
    const stdair::DoWStruct& lDoW = lPeriod.getDoW ();
    if (lDoW.getStandardDayOfWeek (lDay) == false) {
      return false;
    }

    return true;
  }

}
```

24.71 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

24.72 SegmentPathPeriod.hpp

```
#ifndef __AIRSCHED_BOM_SEGMENTPATHPERIOD_HPP
#define __AIRSCHED_BOM_SEGMENTPATHPERIOD_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/BomAbstract.hpp>
// AirSched
#include <airsched/bom/SegmentPathPeriodKey.hpp
>
#include <airsched/bom/SegmentPathPeriodTypes.hpp
>

namespace boost {
  namespace serialization {
    class access;
  }
}
```

```

namespace stdair {
  template <typename BOM> class FacBom;
  class FacBomManager;
  class SegmentPeriod;
}

namespace AIRSCHED {

  class SegmentPathPeriod : public stdair::BomAbstract {
    template <typename BOM> friend class stdair::FacBom;
    friend  class stdair::FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ///////////
    typedef SegmentPathPeriodKey Key_T;


  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    stdair::BomAbstract* const getParent() const {
      return _parent;
    }

    const stdair::PeriodStruct& getDeparturePeriod() const {
      return _key.getPeriod();
    }

    const DateOffsetList_T& getBoardingDateOffsetList
 () const {
      return _key.getBoardingDateOffsetList();
    }

    const stdair::NbOfSegments_T getNbOfSegments() const {
      return _key.getNbOfSegments();
    }

    const stdair::NbOfAirlines_T& getNbOfAirlines() const {
      return _key.getNbOfAirlines();
    }

    const stdair::Duration_T& getElapsedTime() const {
      return _key.getElapsedTime();
    }

    const stdair::Duration_T& getBoardingTime() const {
      return _key.getBoardingTime();
    }

    const stdair::HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    stdair::SegmentPeriod* getLastSegmentPeriod() const;

    stdair::SegmentPeriod* getFirstSegmentPeriod() const;

    const stdair::AirportCode_T& getDestination() const;


  public:
    // ////////////// Business methods ///////////////
    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
;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  protected:
    // ////////// Constructors and destructors /////////
    SegmentPathPeriod (const Key_T&);

    ~SegmentPathPeriod();

  private:
    SegmentPathPeriod();

    SegmentPathPeriod (const SegmentPathPeriod
&);


  protected:
    // ////////// Attributes /////////
    Key_T _key;

    stdair::BomAbstract* _parent;

    stdair::HolderMap_T _holderMap;
  };

}
#endif // __AIRSCHED_BOM_SEGMENTPATHPERIOD_HPP

```

24.73 airsched/bom/SegmentPathPeriodKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <airsched/bom/SegmentPathPeriodKey.hpp>
```

Namespaces

- namespace AIRSCHED

Functions

- template void AIRSCHED::SegmentPathPeriodKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void AIRSCHED::SegmentPathPeriodKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

24.74 SegmentPathPeriodKey.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#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>
// AirSched
#include <airsched/bom/SegmentPathPeriodKey.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  SegmentPathPeriodKey::SegmentPathPeriodKey()
    : _period (stdair::BOOST_DEFAULT_DATE_PERIOD, stdair::DEFAULT_DOW_STRING),
      _boardingTime (stdair::NULL_BOOST_TIME_DURATION),
      _elapsed (stdair::NULL_BOOST_TIME_DURATION),
      _nbOfAirlines (stdair::DEFAULT_NBOFAIRLINES) {
  }

  // //////////////////////////////////////////////////////////////////////
  SegmentPathPeriodKey::SegmentPathPeriodKey (const SegmentPathPeriodKey& iSPPK)
    : _period (iSPPK._period),
      _boardingTime (iSPPK._boardingTime),
      _elapsed (iSPPK._elapsed),
      _boardingDateOffsetList (iSPPK._boardingDateOffsetList),
      _nbOfAirlines (iSPPK._nbOfAirlines) {
  }

  // //////////////////////////////////////////////////////////////////////
  SegmentPathPeriodKey::
  SegmentPathPeriodKey (const stdair::PeriodStruct& iPeriod,
                        const stdair::Duration_T& iBoardingTime,
                        const stdair::Duration_T& iElapsedTime,
                        const DateOffsetList_T& iBoardingDateOffsetList,
                        const stdair::NbOfAirlines_T& iNbOfAirlines)
    : _period (iPeriod),
      _boardingTime (iBoardingTime),
      _elapsed (iElapsedTime),
      _boardingDateOffsetList (iBoardingDateOffsetList),
      _nbOfAirlines (iNbOfAirlines) {
  }

  // //////////////////////////////////////////////////////////////////////
  SegmentPathPeriodKey::~SegmentPathPeriodKey() {
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentPathPeriodKey::toStream (std::ostream& ioOut) const {
    ioOut << "SegmentPathPeriodKey: " << toString() << std::endl;
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentPathPeriodKey::fromStream (std::istream& ioIn) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string SegmentPathPeriodKey::toString() const {
    std::ostringstream oStr;
    oStr << _period.describeShort() << ", "
         << _boardingTime << ", "  << _elapsed << ", ";

    for (DateOffsetList_T::const_iterator itOffset =
           _boardingDateOffsetList.begin();
         itOffset != _boardingDateOffsetList.end(); ++itOffset) {
      const stdair::DateOffset_T& lDateOffset = *itOffset;
      oStr << lDateOffset.days() << ", ";
    }
    oStr << _nbOfAirlines ;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathPeriodKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathPeriodKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void SegmentPathPeriodKey::serialize (Archive&
 ioArchive,
                                        const unsigned int iFileVersion) {
    //ioArchive & _period & _boardingTime & _elapsed & _nbOfAirlines;
    std::string lBTStr = boost::posix_time::to_simple_string (_boardingTime);
    std::string lElapsedStr = boost::posix_time::to_simple_string (_elapsed);
    ioArchive & lBTStr & lElapsedStr & _nbOfAirlines;
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template
  void SegmentPathPeriodKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                           unsigned int);
  template
  void SegmentPathPeriodKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                           unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

24.75 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 AIRSCHED::SegmentPathPeriodKey

  Structure representing the key of a segment/path.

Namespaces

- namespace boost

  Forward declarations.
- namespace boost::serialization
- namespace AIRSCHED

24.76 SegmentPathPeriodKey.hpp

```
#ifndef __AIRSCHED_BOM_SEGMENTPATHPERIODKEY_HPP
```

```cpp
#define __AIRSCHED_BOM_SEGMENTPATHPERIODKEY_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>
// AirSched
#include <airsched/bom/SegmentPathPeriodTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace AIRSCHED {

  struct SegmentPathPeriodKey : public stdair::KeyAbstract {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  public:
    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();


  public:
    // /////////// Getters //////////
    const stdair::PeriodStruct& getPeriod() const {
      return _period;
    }

    const DateOffsetList_T& getBoardingDateOffsetList() const {
      return _boardingDateOffsetList;
    }

    const stdair::NbOfSegments_T getNbOfSegments() const {
      return _boardingDateOffsetList.size();
    }

    const stdair::NbOfAirlines_T& getNbOfAirlines() const {
      return _nbOfAirlines;
    }

    const stdair::Duration_T& getElapsedTime() const {
      return _elapsed;
    }

    const stdair::Duration_T& getBoardingTime() const {
      return _boardingTime;
    }


  public:
    // /////////// Setters //////////
    void setPeriod (const stdair::PeriodStruct& iPeriod) {
      _period = iPeriod;
    }

    void setBoardingDateOffsetList (const DateOffsetList_T& iList) {
      _boardingDateOffsetList = iList;
    }

    void setNbOfAirlines (const stdair::NbOfAirlines_T& iNbOfAirlines) {
      _nbOfAirlines = iNbOfAirlines;
    }

    void setElapsedTime (const stdair::Duration_T& iElapsed) {
      _elapsed = iElapsed;
    }

    void setBoardingTime (const stdair::Duration_T&
 iBoardingTime) {
      _boardingTime = iBoardingTime;
    }


  public:
    // /////////// Business methods ////////////
    const bool isValid () const {
      return _period.isValid ();
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  private:
    // ///////////////// Attributes ///////////////
    stdair::PeriodStruct _period;

    stdair::Duration_T _boardingTime;

    stdair::Duration_T _elapsed;

    DateOffsetList_T _boardingDateOffsetList;

    stdair::NbOfAirlines_T _nbOfAirlines;
  };

}
#endif // __AIRSCHED_BOM_SEGMENTPATHPERIODKEY_HPP
```

24.77 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::SegmentPathPeriodMultimap_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

24.78 SegmentPathPeriodTypes.hpp

```
// //////////////////////////////////////////////////////////////////////
#ifndef __AIRSCHED_BOM_SEGMENTPATHPERIODTYPES_HPP
#define __AIRSCHED_BOM_SEGMENTPATHPERIODTYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <vector>
#include <list>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/key_types.hpp>

namespace AIRSCHED {

  class SegmentPathPeriod;

  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;

}
#endif // __AIRSCHED_BOM_SEGMENTPATHPERIODTYPES_HPP

```

24.79 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

24.80 SegmentPeriodHelper.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
```

```
// STDAIR
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
// AIRSCHED
#include <airsched/bom/SegmentPeriodHelper.hpp
>

namespace AIRSCHED {
  // ////////////////////////////////////////////////////////////////////
  void SegmentPeriodHelper::fill (
stdair::SegmentPeriod& ioSegmentPeriod,
                                  const SegmentStruct&
iSegmentStruct) {
    // Browse the list of segment cabins and fill the cabin booking
    // class map of the BOM segment period.
    for (SegmentCabinStructList_T::const_iterator itCabin =
           iSegmentStruct._cabinList.begin();
         itCabin != iSegmentStruct._cabinList.end(); ++itCabin) {
      const SegmentCabinStruct& lSegmentCabinStruct = *
itCabin;
      ioSegmentPeriod.
        addCabinBookingClassList (lSegmentCabinStruct._cabinCode,
                                  lSegmentCabinStruct._classes);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPeriodHelper::fill (
stdair::SegmentPeriod& ioSegmentPeriod,
                                  const LegStructList_T&
iLegList) {

    const stdair::AirportCode_T& lBoardingPoint =
      ioSegmentPeriod.getBoardingPoint ();
    const stdair::AirportCode_T& lOffPoint = ioSegmentPeriod.getOffPoint();
    stdair::Duration_T lElapsedTime;

    // Find the leg which has the same boarding point.
    LegStructList_T::const_iterator itLeg = iLegList.begin ();
    while (itLeg != iLegList.end()) {
      const LegStruct& lLeg = *itLeg;
      if (lLeg._boardingPoint == lBoardingPoint) {
        break;
      } else {
        ++itLeg;
      }
    }
    assert (itLeg != iLegList.end());
    const LegStruct& lFirstLeg = *itLeg;
    stdair::AirportCode_T lCurrentOffPoint = lFirstLeg._offPoint;
    stdair::Duration_T lCurrentOffTime = lFirstLeg._offTime;

    // Update the elapsed time.
    lElapsedTime += lFirstLeg._elapsed;

    // Find the last used leg.
    while (lCurrentOffPoint != lOffPoint) {
      ++itLeg;
      assert (itLeg != iLegList.end());

      const LegStruct& lCurrentLeg = *itLeg;
      assert (lCurrentOffPoint == lCurrentLeg._boardingPoint);
      // As the boarding point of the current leg is the same as the off point
      // of the previous leg (by construction), there is no time difference.
      const stdair::Duration_T lStopOverTime =
        lCurrentLeg._boardingTime - lCurrentOffTime;
      lElapsedTime += lStopOverTime;

      // Add the elapsed time of the current leg
      lElapsedTime += lCurrentLeg._elapsed;

      lCurrentOffTime = lCurrentLeg._offTime;
      lCurrentOffPoint = lCurrentLeg._offPoint;
    }
    const LegStruct& lLastLeg = *itLeg;

    // Update the attributes of the segment-period.
    ioSegmentPeriod.setBoardingTime (lFirstLeg._boardingTime);
    ioSegmentPeriod.setOffTime (lLastLeg._offTime);
    ioSegmentPeriod.setBoardingDateOffset (lFirstLeg._boardingDateOffset);
    ioSegmentPeriod.setOffDateOffset (lLastLeg._offDateOffset);
    ioSegmentPeriod.setElapsedTime (lElapsedTime);
  }

}
```

24.81 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

24.82 SegmentPeriodHelper.hpp

```
#ifndef __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP
#define __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// AIRSCHED
#include <airsched/bom/LegStruct.hpp>
#include <airsched/bom/SegmentStruct.hpp>

// Forward declarations
namespace stdair {
  class SegmentPeriod;
}

namespace AIRSCHED {
  class SegmentPeriodHelper {
  public:
    // ////////// Business Methods /////////
    static void fill (stdair::SegmentPeriod&, const SegmentStruct
&);

    static void fill (stdair::SegmentPeriod&, const LegStructList_T
&);
  };

}
#endif // __AIRSCHED_BOM_SEGMENTPERIODHELPER_HPP
```

24.83 airsched/bom/SegmentStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentDate.hpp>
#include <airsched/bom/SegmentStruct.hpp>
```

Namespaces

- namespace AIRSCHED

24.84 SegmentStruct.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// STDAIR
#include <stdair/bom/SegmentDate.hpp>
// AIRSCHED
#include <airsched/bom/SegmentStruct.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  const std::string SegmentStruct::describe() const {
    std::ostringstream ostr;
    ostr << "    " << _boardingPoint << " / "
         << boost::posix_time::to_simple_string(_boardingTime)
         << " -- " << _offPoint << " / "
         << boost::posix_time::to_simple_string(_offTime)
         << " --> "
         << boost::posix_time::to_simple_string(_elapsed)
         << std::endl;
    for (SegmentCabinStructList_T::const_iterator itCabin =
           _cabinList.begin(); itCabin != _cabinList.end();
 itCabin++) {
      const SegmentCabinStruct& lCabin = *itCabin;
      ostr << lCabin.describe();
    }
    ostr << std::endl;

    return ostr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentStruct::fill (stdair::SegmentDate&
ioSegmentDate) const {
    // Note that some parameters (boarding date, boarding time, off
    // date, off time, elapsed time) are set by
    // SegmentDate::fillFromRouting() when the routing (with legs) is
    // built. So, it is useless to set those parameters here.

    // At that time, there are no other parameters.
  }

}
```

24.85 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

24.86 SegmentStruct.hpp

```
#ifndef __AIRSCHED_BOM_SEGMENTSTRUCT_HPP
#define __AIRSCHED_BOM_SEGMENTSTRUCT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
// AirSched
#include <airsched/bom/SegmentCabinStruct.hpp
>

// Forward declarations
namespace stdair {
  class SegmentDate;
}

namespace AIRSCHED {

  struct SegmentStruct : public stdair::StructAbstract {
    // 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;

    void fill (stdair::SegmentDate&) const;

    const std::string describe() const;
  };

  typedef std::vector<SegmentStruct> SegmentStructList_T;

}
#endif // __AIRSCHED_BOM_SEGMENTSTRUCT_HPP
```

24.87 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

24.88 InventoryGenerator.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// Boost
#include <boost/date_time/date_iterator.hpp>
// StdAir
#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>
// AirSched
#include <airsched/bom/FlightPeriodStruct.hpp>
#include <airsched/bom/SegmentPeriodHelper.hpp>
#include <airsched/command/InventoryGenerator.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  void InventoryGenerator::
  createFlightPeriod (stdair::BomRoot& ioBomRoot,
                      const FlightPeriodStruct& iFlightPeriodStruct) {

    const stdair::AirlineCode_T& lAirlineCode = iFlightPeriodStruct._airlineCode;

    // Instantiate an inventory object (if not exist)
    // for the given key (airline code)
    stdair::Inventory* lInventory_ptr = stdair::BomManager::
      getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
    if (lInventory_ptr == NULL) {
      stdair::InventoryKey lKey (lAirlineCode);

      lInventory_ptr =
        &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
      stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
      stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
    }
    assert (lInventory_ptr != NULL);

    // Create the flight-period key.
    const stdair::PeriodStruct lPeriod (iFlightPeriodStruct._dateRange,
                                        iFlightPeriodStruct._dow);
    const stdair::FlightPeriodKey
      lFlightPeriodKey (iFlightPeriodStruct._flightNumber, lPeriod);

    // Check that the flight-period object is not already created.
    stdair::FlightPeriod* lFlightPeriod_ptr = stdair::BomManager::
      getObjectPtr<stdair::FlightPeriod> (*lInventory_ptr,
                                          lFlightPeriodKey.toString());
    if (lFlightPeriod_ptr != NULL) {
      throw stdair::ObjectCreationgDuplicationException ("");
    }
    assert (lFlightPeriod_ptr == NULL);

    // Instantiate a flight-period object with the given key.
    lFlightPeriod_ptr = &stdair::FacBom<stdair::FlightPeriod>::
      instance().create (lFlightPeriodKey);
    stdair::FacBomManager::addToListAndMap (*lInventory_ptr, *lFlightPeriod_ptr);
    stdair::FacBomManager::linkWithParent (*lInventory_ptr, *lFlightPeriod_ptr);

    // Create the segment-periods.
    createSegmentPeriods (*lFlightPeriod_ptr, iFlightPeriodStruct);
  }

  // //////////////////////////////////////////////////////////////////////
  void InventoryGenerator::
  createSegmentPeriods (stdair::FlightPeriod& ioFlightPeriod,
                        const FlightPeriodStruct& iFlightPeriodStruct) {

    // Iterate on the segment strutures.
    const SegmentStructList_T& lSegmentList = iFlightPeriodStruct._segmentList;
    for (SegmentStructList_T::const_iterator itSegment = lSegmentList.begin();
         itSegment != lSegmentList.end(); ++itSegment) {

      const SegmentStruct& lSegment = *itSegment;

      // Set the segment-period primary key.
      const stdair::AirportCode_T& lBoardingPoint = lSegment._boardingPoint;
      const stdair::AirportCode_T& lOffPoint = lSegment._offPoint;
      const stdair::SegmentPeriodKey lSegmentPeriodKey (lBoardingPoint,
                                                        lOffPoint);

      // Instantiate a segment-perioed with the key.
      stdair::SegmentPeriod& lSegmentPeriod = stdair::
        FacBom<stdair::SegmentPeriod>::instance().create (lSegmentPeriodKey);
      stdair::FacBomManager::addToListAndMap (ioFlightPeriod, lSegmentPeriod);
      stdair::FacBomManager::linkWithParent (ioFlightPeriod, lSegmentPeriod);

      // Set the segment-period attributes.
      SegmentPeriodHelper::fill (lSegmentPeriod,
lSegment);
      SegmentPeriodHelper::fill (lSegmentPeriod,
iFlightPeriodStruct._legList);
    }
  }

}
```

24.89 airsched/command/InventoryGenerator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHED_Types.hpp>
```

Classes

- class AIRSCHED::InventoryGenerator

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRSCHED
- namespace AIRSCHED::ScheduleParserHelper

24.90 InventoryGenerator.hpp

```
#ifndef __AIRSCHED_CMD_INVENTORYGENERATOR_HPP
#define __AIRSCHED_CMD_INVENTORYGENERATOR_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/command/CmdAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>

// Forward declarations
namespace stdair {
  class BomRoot;
  class FlightPeriod;
}

namespace AIRSCHED {

  // Forward declarations
  struct FlightPeriodStruct;
  struct LegStruct;
  struct SegmentStruct;
  struct LegCabinStruct;
  struct SegmentCabinStruct;
  namespace ScheduleParserHelper {
    struct doEndFlight;
  }

  class InventoryGenerator : public stdair::CmdAbstract {
    // Only the following class may use methods of InventoryGenerator.
    // Indeed, as those methods build the BOM, it is not good to expose
    // them publicly.
    friend class FlightPeriodFileParser;
    friend class FFFlightPeriodFileParser;
    friend struct ScheduleParserHelper::doEndFlight
;
    friend class ScheduleParser;

  private:
    static void createFlightPeriod (stdair::BomRoot&,
                                    const FlightPeriodStruct&
);

    static void createSegmentPeriods (stdair::FlightPeriod&,
                                      const FlightPeriodStruct
&);

  };

}
#endif // __AIRSCHED_CMD_INVENTORYGENERATOR_HPP
```

24.91 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

24.92 OnDParser.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
// AirSched
#include <airsched/command/OnDParserHelper.hpp
>
#include <airsched/command/OnDParser.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  void OnDParser::generateOnDPeriods (const
stdair::Filename_T& iFilename,
                                      stdair::BomRoot& ioBomRoot) {

    // Check that the file path given as input corresponds to an actual file
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (iFilename);

    if (doesExistAndIsReadable == false) {
      throw OnDInputFileNotFoundException ("The
 O&D file " + iFilename
                                           + " does not exist or can not be "
                                           "read");
    }

    // Initialise the O&D-Period file parser.
    OnDPeriodFileParser lOnDPeriodParser (iFilename,
ioBomRoot);

    // Parse the CSV-formatted O&D input file, and generate the
    // corresponding O&D-Period for the airlines.
    lOnDPeriodParser.generateOnDPeriods();
  }

}
```

24.93 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

24.94 OnDParser.hpp

```
#ifndef __AIRSCHED_CMD_ONDPARSER_HPP
#define __AIRSCHED_CMD_ONDPARSER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  class OnDParser : public stdair::CmdAbstract {
  public:
    static void generateOnDPeriods (const stdair::Filename_T&
,
                                    stdair::BomRoot&);
  };

}
#endif // __AIRSCHED_CMD_ONDPARSER_HPP
```

24.95 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

24.96 OnDParserHelper.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
// AIRSCHED
#include <airsched/command/OnDParserHelper.hpp
>
#include <airsched/command/OnDPeriodGenerator.hpp
>

namespace AIRSCHED {

  namespace OnDParserHelper {

    // //////////////////////////////////////////////////////////////////////
    //
    //  Semantic actions
    //
    // //////////////////////////////////////////////////////////////////////

    ParserSemanticAction::
    ParserSemanticAction (OnDPeriodStruct
& ioOnDPeriod)
      : _onDPeriod (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////////
    storeOrigin::storeOrigin (OnDPeriodStruct
& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////////
    void storeOrigin::operator() (iterator_t
iStr,
                                  iterator_t iStrEnd) const {
      std::string lOrigin (iStr, iStrEnd);
      //STDAIR_LOG_DEBUG ( "Origin: " << lOrigin << std::endl);

      // Set the origin
      _onDPeriod._origin = lOrigin;
      _onDPeriod._nbOfAirlines = 0;
      _onDPeriod._airlineCode = "";
      _onDPeriod._classCode = "";
      _onDPeriod._airlineCodeList.clear();
      _onDPeriod._classCodeList.clear();
    }

    // //////////////////////////////////////////////////////////////////
    storeDestination::storeDestination (
OnDPeriodStruct& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeDestination::operator() (iterator_t
 iStr,
                                       iterator_t iStrEnd) const {
      std::string lDestination (iStr, iStrEnd);
      //STDAIR_LOG_DEBUG ("Destination: " << lDestination << std::endl);

      // Set the destination
      _onDPeriod._destination = lDestination;
    }

    // //////////////////////////////////////////////////////////////////
    storeDateRangeStart::
    storeDateRangeStart (OnDPeriodStruct&
 ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeDateRangeStart::operator() (
iterator_t iStr,
                                          iterator_t iStrEnd) const {
      _onDPeriod._dateRangeStart = _onDPeriod
.getDate();
      /*STDAIR_LOG_DEBUG ("Date Range Start: "
        << _onDPeriod._dateRangeStart << std::endl);*/

      // Reset the number of seconds
      _onDPeriod._itSeconds = 0;
    }

    // //////////////////////////////////////////////////////////////////
    storeDateRangeEnd::
    storeDateRangeEnd (OnDPeriodStruct&
ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeDateRangeEnd::operator() (
iterator_t iStr,
                                        iterator_t iStrEnd) const {
      // As a Boost date period (COM::DatePeriod_T) defines the last day of
      // the period to be end-date - one day, we have to add one day to that
      // end date before.
      const stdair::DateOffset_T oneDay (1);
      _onDPeriod._dateRangeEnd = _onDPeriod.
getDate() + oneDay;
      /*STDAIR_LOG_DEBUG ( "Date Range End: "
        << _onDPeriod._dateRangeEnd << std::endl);*/

      // Transform the date pair (i.e., the date range) into a date period
      _onDPeriod._datePeriod =
        stdair::DatePeriod_T (_onDPeriod._dateRangeStart
,
                              _onDPeriod._dateRangeEnd);

      // Reset the number of seconds
      _onDPeriod._itSeconds = 0;
    }

    // //////////////////////////////////////////////////////////////////
    storeStartRangeTime::
    storeStartRangeTime (OnDPeriodStruct&
 ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeStartRangeTime::operator() (
iterator_t iStr,
                                          iterator_t iStrEnd) const {
```

```
      _onDPeriod._timeRangeStart = _onDPeriod.getTime();

      // Reset the number of seconds
      _onDPeriod._itSeconds = 0;
    }

    // //////////////////////////////////////////////////////////////////
    storeEndRangeTime::
    storeEndRangeTime (OnDPeriodStruct& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeEndRangeTime::operator() (iterator_t iStr,
                                        iterator_t iStrEnd) const {
      _onDPeriod._timeRangeEnd = _onDPeriod.getTime();

      // Reset the number of seconds
      _onDPeriod._itSeconds = 0;
    }

    // //////////////////////////////////////////////////////////////////
    storeAirlineCode::
    storeAirlineCode (OnDPeriodStruct& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeAirlineCode::operator() (iterator_t iStr,
                                       iterator_t iStrEnd) const {
      const std::string lAirlineCodeStr (iStr, iStrEnd);
      const stdair::AirlineCode_T lAirlineCode(lAirlineCodeStr);
      // Test if the OnD Period Struct stands for interline products
      if (_onDPeriod._airlineCodeList.size() > 0) {
        // update the airline code
        std::ostringstream ostr;
        ostr << _onDPeriod._airlineCode << lAirlineCode;
        _onDPeriod._airlineCode = ostr.str();
        // Update the number of airlines if necessary
        const stdair::AirlineCode_T lPreviousAirlineCode =
          _onDPeriod._airlineCodeList.back();
        if (lPreviousAirlineCode != lAirlineCode) {
          _onDPeriod._nbOfAirlines = _onDPeriod._nbOfAirlines + 1;
        }
      }
      else {
        _onDPeriod._airlineCode = lAirlineCode;
        _onDPeriod._nbOfAirlines = 1;
      }
      _onDPeriod._airlineCodeList.push_back (lAirlineCode);

      //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode << std::endl);
    }

    // //////////////////////////////////////////////////////////////////
    storeClassCode::
    storeClassCode (OnDPeriodStruct& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeClassCode::operator() (char iChar) const {
      std::ostringstream ostr;
      ostr << iChar;
      std::string classCodeStr = ostr.str();
      const stdair::ClassCode_T lClassCode (classCodeStr);
      _onDPeriod._classCodeList.push_back(lClassCode);
      /*STDAIR_LOG_DEBUG ("Class Code: "
        << lClassCode << std::endl);*/
      // Insertion of this class Code in the whole classCode name
      std::ostringstream ostrr;
      ostrr << _onDPeriod._classCode << classCodeStr;
      _onDPeriod._classCode = ostrr.str();

    }

    // //////////////////////////////////////////////////////////////////
    doEndOnD::doEndOnD (stdair::BomRoot& ioBomRoot,
OnDPeriodStruct& ioOnDPeriod)
      : ParserSemanticAction (ioOnDPeriod),
        _bomRoot (ioBomRoot) {
    }

    // //////////////////////////////////////////////////////////////////
    void doEndOnD::operator() (iterator_t iStr,
iterator_t iStrEnd) const {

      // DEBUG: Display the result
      // STDAIR_LOG_DEBUG ("FareRule " << _onDPeriod.describe());

      // Generation of the O&D-Period object.
      OnDPeriodGenerator::createOnDPeriod (_bomRoot, _onDPeriod
);
    }

    // //////////////////////////////////////////////////////////////////
    //
    //  Utility Parsers
    //
    // //////////////////////////////////////////////////////////////////

    uint2_p_t uint2_p;

    uint4_p_t uint4_p;

    uint1_4_p_t uint1_4_p;

    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");

    //
    //  (Boost Spirit) Grammar Definition
    //

    // //////////////////////////////////////////////////////////////////
    OnDParser::
    OnDParser (stdair::BomRoot& ioBomRoot, OnDPeriodStruct
& ioOnDPeriod)
      : _bomRoot (ioBomRoot), _onDPeriod (ioOnDPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    template<typename ScannerT>
    OnDParser::definition<ScannerT>::definition
 (OnDParser const& self) {

      ond_list = *( boost::spirit::classic::comment_p("//")
                    | boost::spirit::classic::comment_p("/*", "*/")
                    | ond )
        ;

      ond = ond_key
        >> +( ';' >> segment )
        >> ond_end[doEndOnD(self._bomRoot, self._onDPeriod)]
        ;

      ond_end = boost::spirit::classic::ch_p(';')
        ;

      ond_key = (airport_p)[storeOrigin(self._onDPeriod)]
```

```
        >> ';' >> (airport_p)[storeDestination(self.
_onDPeriod)]
        >> ';' >> date[storeDateRangeStart(self._onDPeriod)]
        >> ';' >> date[storeDateRangeEnd(self._onDPeriod)]
        >> ';' >> time[storeStartRangeTime(self._onDPeriod)]
        >> ';' >> time[storeEndRangeTime(self._onDPeriod)]
        ;

      date = boost::spirit::classic::
        lexeme_d[(year_p)[boost::spirit::classic::
                          assign_a(self._onDPeriod._itYear)]
                 >> '-'
                 >> (month_p)[boost::spirit::classic::
                              assign_a(self._onDPeriod._itMonth)]
                 >> '-'
                 >> (day_p)[boost::spirit::classic::
                            assign_a(self._onDPeriod._itDay)]]
        ;

      time = boost::spirit::classic::
        lexeme_d[(hours_p)[boost::spirit::classic::
                           assign_a(self._onDPeriod._itHours)]
                 >> ':'
                 >> (minutes_p)[boost::spirit::classic::
                                assign_a(self._onDPeriod._itMinutes)]
                 >> !(':' >> (seconds_p)[boost::spirit::classic::
                                         assign_a(self._onDPeriod._itSeconds)])
]
        ;

      segment = boost::spirit::classic::
        lexeme_d[(airline_code_p)[storeAirlineCode
(self._onDPeriod)]]
        >> ';' >> (class_code_p)[storeClassCode(self.
_onDPeriod)]
        ;

      //BOOST_SPIRIT_DEBUG_NODE (OnDParser);
      BOOST_SPIRIT_DEBUG_NODE (ond_list);
      BOOST_SPIRIT_DEBUG_NODE (ond);
      BOOST_SPIRIT_DEBUG_NODE (segment);
      BOOST_SPIRIT_DEBUG_NODE (ond_key);
      BOOST_SPIRIT_DEBUG_NODE (ond_end);
      BOOST_SPIRIT_DEBUG_NODE (date);
      BOOST_SPIRIT_DEBUG_NODE (time);

    }

    // //////////////////////////////////////////////////////////////////////
    template<typename ScannerT>
    boost::spirit::classic::rule<ScannerT> const&
    OnDParser::definition<ScannerT>::start
() const {
      return ond_list;
    }
  }

  //
  //  Entry class for the file parser
  //

  // //////////////////////////////////////////////////////////////////////
  OnDPeriodFileParser::OnDPeriodFileParser
 (const stdair::Filename_T& iFilename,
                                            stdair::BomRoot& ioBomRoot)
    : _filename (iFilename), _bomRoot (ioBomRoot) {
    init();
  }

  // //////////////////////////////////////////////////////////////////////
  void OnDPeriodFileParser::init() {
    // Check that the file exists and is readable
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (_filename);

    if (doesExistAndIsReadable == false) {
      STDAIR_LOG_ERROR ("The O&D file " << _filename
                        << " does not exist or can not be read.");

      throw OnDInputFileNotFoundException ("The
 O&D file " + _filename
                                           + " does not exist or can not be
 read");
    }

    // Open the file
    _startIterator = iterator_t (_filename);

    // Check that the filename exists and can be open
    if (!_startIterator) {
      STDAIR_LOG_DEBUG ("The O&D file " << _filename << " can not be open."
                          << std::endl);
      throw OnDInputFileNotFoundException ("The file " + _filename
                                           + " does not exist or can not be
 read");
    }

    // Create an EOF iterator
    _endIterator = _startIterator.make_end();
  }

  // //////////////////////////////////////////////////////////////////////
  bool OnDPeriodFileParser::generateOnDPeriods
 () {
    bool oResult = false;

    STDAIR_LOG_DEBUG ("Parsing O&D input file: " << _filename);

    // Initialise the parser (grammar) with the helper/staging structure.
    OnDParserHelper::OnDParser lODParser (_bomRoot,
_onDPeriod);

    // Launch the parsing of the file and, thanks to the doEndOnD
    // call-back structure, filling the worldSchedule (Fares)
    boost::spirit::classic::parse_info<iterator_t> info =
      boost::spirit::classic::parse (_startIterator, _endIterator, lODParser,
                                     boost::spirit::classic::space_p);

    // Retrieves whether or not the parsing was successful
    oResult = info.hit;

    const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
    if (oResult == true) {
      STDAIR_LOG_DEBUG ("Parsing of O&D input file: " << _filename
                        << " succeeded: read " << info.length
                        << " characters. The input file has "
                        << hasBeenFullyReadStr
                        << "been fully read. Stop point: " << info.stop);

    } else {
      // TODO: decide whether to throw an exception
      STDAIR_LOG_ERROR ("Parsing of O&D input file: " << _filename
                        << " failed: read " << info.length
                        << " characters. The input file has "
                        << hasBeenFullyReadStr
                        << "been fully read. Stop point: " << info.stop);
    }

    return oResult;
  }
}
```

24.97 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/AIRSCHED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/OnDPeriodStruct.hpp>
```

Classes

- struct AIRSCHED::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

24.98 OnDParserHelper.hpp

```
#ifndef __AIRSCHED_CMD_ONDPARSERHELPER_HPP
#define __AIRSCHED_CMD_ONDPARSERHELPER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost (Extended STL)
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
// StdAir
#include <stdair/command/CmdAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/OnDPeriodStruct.hpp>

// Forward declarations
namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  namespace OnDParserHelper {

    // //////////////////////////////////////////////////////////////////////
    //
    //  Semantic actions
    //
    // //////////////////////////////////////////////////////////////////////
    struct ParserSemanticAction {
      ParserSemanticAction (OnDPeriodStruct&
);
      OnDPeriodStruct& _onDPeriod;
    };

    struct storeOrigin : public ParserSemanticAction
 {
      storeOrigin (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeDestination : public ParserSemanticAction
 {
      storeDestination (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeDateRangeStart : public ParserSemanticAction
 {
      storeDateRangeStart (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeDateRangeEnd : public ParserSemanticAction
  {
      storeDateRangeEnd (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
  iStrEnd) const;
    };

    struct storeStartRangeTime : public ParserSemanticAction
  {
      storeStartRangeTime (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
  iStrEnd) const;
    };

    struct storeEndRangeTime : public ParserSemanticAction
  {
      storeEndRangeTime (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
  iStrEnd) const;
    };

    struct storeAirlineCode : public ParserSemanticAction
  {
      storeAirlineCode (OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
  iStrEnd) const;
    };

    struct storeClassCode : public ParserSemanticAction
  {
      storeClassCode (OnDPeriodStruct&);
      void operator() (char iChar) const;
    };

    struct doEndOnD : public ParserSemanticAction {
      doEndOnD (stdair::BomRoot&, OnDPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
  iStrEnd) const;
      stdair::BomRoot& _bomRoot;
    };

    //
    //  (Boost Spirit) Grammar Definition
    //

    struct OnDParser :
      public boost::spirit::classic::grammar<OnDParser> {

      OnDParser (stdair::BomRoot&, OnDPeriodStruct&);

      template <typename ScannerT>
      struct definition {
        definition (OnDParser const& self);

        // Instantiation of rules
        boost::spirit::classic::rule<ScannerT> ond_list, ond,
 segment,
          ond_key, ond_end, date, time;

        boost::spirit::classic::rule<ScannerT> const& start() const;
      };

      // Parser Context
      stdair::BomRoot& _bomRoot;
      OnDPeriodStruct& _onDPeriod;
    };
  }

  //
  //  Entry class for the file parser
  //

  class OnDPeriodFileParser : public stdair::CmdAbstract {
  public:
    OnDPeriodFileParser (const stdair::Filename_T& iFilename
,
                         stdair::BomRoot& ioBomRoot);

    bool generateOnDPeriods ();

  private:
    void init();

  private:
    // Attributes
    stdair::Filename_T _filename;

    iterator_t _startIterator;

    iterator_t _endIterator;

    stdair::BomRoot& _bomRoot;

    OnDPeriodStruct _onDPeriod;
  };

}
#endif // __AIRSCHED_CMD_ONDPARSERHELPER_HPP
```

24.99 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

24.100 OnDPeriodGenerator.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#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>
// AirSched
#include <airsched/bom/OnDPeriodStruct.hpp>
#include <airsched/command/OnDPeriodGenerator.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  void OnDPeriodGenerator::
  createOnDPeriod (stdair::BomRoot& ioBomRoot,
                   const OnDPeriodStruct& iOnDPeriodStruct) {
  }
}
```

24.101 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

24.102 OnDPeriodGenerator.hpp

```
#ifndef __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP
#define __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/command/CmdAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>

namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  struct OnDPeriodStruct_T;
  namespace OnDParserHelper {
    struct doEndOnD;
  }

  class OnDPeriodGenerator : public stdair::CmdAbstract {
    friend class OnDPeriodFileParser;
    friend struct OnDParserHelper::doEndOnD;
    friend class OnDParser;

  private:
    static void createOnDPeriod (stdair::BomRoot&, const OnDPeriodStruct
&);
  };

}
#endif // __AIRSCHED_CMD_ONDPERIODGENERATOR_HPP
```

24.103 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

24.104 ScheduleParser.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
// StdAir
```

```
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
// AirSched
#include <airsched/command/SegmentPathGenerator.hpp
>
#include <airsched/command/ScheduleParserHelper.hpp
>
#include <airsched/command/ScheduleParser.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  void ScheduleParser::generateInventories (
const stdair::Filename_T& iFilename,
                                            stdair::BomRoot& ioBomRoot) {

    // Check that the file path given as input corresponds to an actual file
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (iFilename);

    if (doesExistAndIsReadable == false) {
      throw ScheduleInputFileNotFoundException
 ("The schedule file " + iFilename
                                                + " does not exist or can not "
                                                "be read");
    }

    // Initialise the Flight-Period file parser.
    FlightPeriodFileParser lFlightPeriodParser (ioBomRoot
, iFilename);

    // Parse the CSV-formatted schedule input file, and generate the
    // corresponding Inventories for the airlines.
    lFlightPeriodParser.generateInventories();

    // Build the network from the schedule.
    SegmentPathGenerator::createSegmentPathNetwork
 (ioBomRoot);
  }

}
```

24.105 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

24.106 ScheduleParser.hpp

```
#ifndef __AIRSCHED_CMD_SCHEDULEPARSER_HPP
#define __AIRSCHED_CMD_SCHEDULEPARSER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
```

```
#include <stdair/command/CmdAbstract.hpp>

// Forward declarations.
namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  class ScheduleParser : public stdair::CmdAbstract {
  public:
    static void generateInventories (const
stdair::Filename_T&,
                                     stdair::BomRoot&);
  };
}
#endif // __AIRSCHED_CMD_SCHEDULEPARSER_HPP
```

24.107 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

24.108 ScheduleParserHelper.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
// AIRSCHED
//#define BOOST_SPIRIT_DEBUG
#include <airsched/command/ScheduleParserHelper.hpp
>
#include <airsched/command/InventoryGenerator.hpp
>

namespace bsc = boost::spirit::classic;

namespace AIRSCHED {

  namespace ScheduleParserHelper {

    // ////////////////////////////////////////////////////////////////////
    //  Semantic actions
    // ////////////////////////////////////////////////////////////////////

    ParserSemanticAction::
    ParserSemanticAction (FlightPeriodStruct
& ioFlightPeriod)
      : _flightPeriod (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    storeAirlineCode::
    storeAirlineCode (FlightPeriodStruct&
 ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeAirlineCode::operator() (iterator_t
 iStr,
                                        iterator_t iStrEnd) const {
      const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
      _flightPeriod._airlineCode = lAirlineCode;

      // As that’s the beginning of a new flight, the list of legs
      // must be reset
      _flightPeriod._legList.clear();
    }

    // ////////////////////////////////////////////////////////////////////
    storeFlightNumber::
    storeFlightNumber (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeFlightNumber::operator() (unsigned
int iNumber) const {
      _flightPeriod._flightNumber = iNumber;
    }

    // ////////////////////////////////////////////////////////////////////
    storeDateRangeStart::
    storeDateRangeStart (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeDateRangeStart::operator() (
iterator_t iStr,
                                           iterator_t iStrEnd) const {
      _flightPeriod._dateRangeStart = _flightPeriod
.getDate();

      // Reset the number of seconds
      _flightPeriod._itSeconds = 0;
    }

    // ////////////////////////////////////////////////////////////////////
    storeDateRangeEnd::
    storeDateRangeEnd (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeDateRangeEnd::operator() (
iterator_t iStr,
                                        iterator_t iStrEnd) const {
      // As a Boost date period (DatePeriod_T) defines the last day of
      // the period to be end-date - one day, we have to add one day to that
      // end date before.
      const stdair::DateOffset_T oneDay (1);
      _flightPeriod._dateRangeEnd = _flightPeriod
.getDate() + oneDay;

      // Transform the date pair (i.e., the date range) into a date period
      _flightPeriod._dateRange =
        stdair::DatePeriod_T (_flightPeriod._dateRangeStart
,
                              _flightPeriod._dateRangeEnd
);

      // Reset the number of seconds
      _flightPeriod._itSeconds = 0;
    }

    // //////////////////////////////////////////////////////////////////
    storeDow::storeDow (FlightPeriodStruct&
 ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeDow::operator() (iterator_t iStr,
iterator_t iStrEnd) const {
      stdair::DOW_String_T lDow (iStr, iStrEnd);
      _flightPeriod._dow = lDow;
    }

    // //////////////////////////////////////////////////////////////////
    storeLegBoardingPoint::
    storeLegBoardingPoint (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeLegBoardingPoint::operator() (
iterator_t iStr,
                                            iterator_t iStrEnd) const
 {
      stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);

      // If a leg has already been parsed, add it to the FlightPeriod
      if (_flightPeriod._legAlreadyDefined ==
true) {
        _flightPeriod._legList.push_back (_flightPeriod
._itLeg);
      } else {
        _flightPeriod._legAlreadyDefined = true;
      }

      // Set the (new) boarding point
      _flightPeriod._itLeg._boardingPoint =
lBoardingPoint;

      // As that's the beginning of a new leg, the list of cabins
      // must be reset
      _flightPeriod._itLeg._cabinList.clear();

      // Add the airport code if it is not already stored in the airport lists
      _flightPeriod.addAirport (lBoardingPoint);
    }

    // //////////////////////////////////////////////////////////////////
    storeLegOffPoint::
    storeLegOffPoint (FlightPeriodStruct&
 ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeLegOffPoint::operator() (iterator_t
 iStr,
                                       iterator_t iStrEnd) const {
      stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
      _flightPeriod._itLeg._offPoint = lOffPoint;

      // Add the airport code if it is not already stored in the airport lists
      _flightPeriod.addAirport (lOffPoint);
    }

    // ////////////////////////////////////////////////////////////////////
    storeBoardingTime::
    storeBoardingTime (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeBoardingTime::operator() (
iterator_t iStr,
                                        iterator_t iStrEnd) const {
      _flightPeriod._itLeg._boardingTime =
_flightPeriod.getTime();

      // Reset the number of seconds
      _flightPeriod._itSeconds = 0;

      // Reset the date off-set
      _flightPeriod._dateOffset = 0;
    }

    // ////////////////////////////////////////////////////////////////////
    storeOffTime::
    storeOffTime (FlightPeriodStruct&
ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeOffTime::operator() (iterator_t
 iStr,
                                   iterator_t iStrEnd) const {
      _flightPeriod._itLeg._offTime = _flightPeriod
.getTime();

      // Reset the number of seconds
      _flightPeriod._itSeconds = 0;

      // As the boarding date off set is optional, it can be set only
      // afterwards, based on the staging date off-set value
      // (_flightPeriod._dateOffset).
      const stdair::DateOffset_T lDateOffset (_flightPeriod.
_dateOffset);
      _flightPeriod._itLeg._boardingDateOffset
 = lDateOffset;
    }

    // ////////////////////////////////////////////////////////////////////
    storeElapsedTime::
    storeElapsedTime (FlightPeriodStruct&
 ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeElapsedTime::operator() (iterator_t
 iStr,
                                       iterator_t iStrEnd) const {
      _flightPeriod._itLeg._elapsed = _flightPeriod
.getTime();

      // Reset the number of seconds
      _flightPeriod._itSeconds = 0;

      // As the boarding date off set is optional, it can be set only
      // afterwards, based on the staging date off-set value
      // (_flightPeriod._dateOffset).
      const stdair::DateOffset_T lDateOffset (_flightPeriod.
_dateOffset);
      _flightPeriod._itLeg._offDateOffset =
lDateOffset;
    }

    // ////////////////////////////////////////////////////////////////////
    storeLegCabinCode::
    storeLegCabinCode (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // ////////////////////////////////////////////////////////////////////
    void storeLegCabinCode::operator() (char
iChar) const {
      _flightPeriod._itLegCabin._cabinCode =
iChar;
      //std::cout << "Cabin code: " << iChar << std::endl;
    }

    // //////////////////////////////////////////////////////////////////
    storeCapacity::
    storeCapacity (FlightPeriodStruct&
ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeCapacity::operator() (double iReal)
 const {
      _flightPeriod._itLegCabin._capacity =
iReal;
      //std::cout << "Capacity: " << iReal << std::endl;

      // The capacity is the last (according to the arrival order
      // within the schedule input file) detail of the leg cabin. Hence,
      // when a capacity is parsed, it means that the full cabin
      // details have already been parsed as well: the cabin can
      // thus be added to the leg.
      _flightPeriod._itLeg._cabinList.push_back (
_flightPeriod._itLegCabin);
    }

    // //////////////////////////////////////////////////////////////////
    storeSegmentSpecificity::
    storeSegmentSpecificity (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeSegmentSpecificity::operator()
(char iChar) const {
      if (iChar == '0') {
        _flightPeriod._areSegmentDefinitionsSpecific
 = false;
      } else {
        _flightPeriod._areSegmentDefinitionsSpecific
 = true;
      }

      // Do a few sanity checks: the two lists should get exactly the same
      // content (in terms of airport codes). The only difference is that one
      // is a STL set, and the other a STL vector.
      assert (_flightPeriod._airportList.size()
              == _flightPeriod._airportOrderedList
.size());
      assert (_flightPeriod._airportList.size() >= 2);

      // Since all the legs have now been parsed, we get all the airports
      // and the segments may be built.
      _flightPeriod.buildSegments();
    }

    // //////////////////////////////////////////////////////////////////
    storeSegmentBoardingPoint::
    storeSegmentBoardingPoint (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeSegmentBoardingPoint::operator()
(iterator_t iStr,
                                                      iterator_t iStrEnd)
 const {
      stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
      _flightPeriod._itSegment._boardingPoint
 = lBoardingPoint;
    }

    // //////////////////////////////////////////////////////////////////
    storeSegmentOffPoint::
    storeSegmentOffPoint (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeSegmentOffPoint::operator() (
iterator_t iStr,
                                             iterator_t iStrEnd) const
{
      stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
      _flightPeriod._itSegment._offPoint =
lOffPoint;
    }

    // //////////////////////////////////////////////////////////////////
    storeSegmentCabinCode::
    storeSegmentCabinCode (FlightPeriodStruct
& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeSegmentCabinCode::operator() (
char iChar) const {
      _flightPeriod._itSegmentCabin._cabinCode
 = iChar;
    }

    // //////////////////////////////////////////////////////////////////
    storeClasses::
    storeClasses (FlightPeriodStruct&
ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeClasses::operator() (iterator_t
 iStr,
                                    iterator_t iStrEnd) const {
      std::string lClasses (iStr, iStrEnd);
      _flightPeriod._itSegmentCabin._classes
 = lClasses;

      // The list of classes is the last (according to the arrival order
      // within the schedule input file) detail of the segment cabin. Hence,
      // when a list of classes is parsed, it means that the full segment
      // cabin details have already been parsed as well: the segment cabin
      // can thus be added to the segment.
      if (_flightPeriod._areSegmentDefinitionsSpecific
 == true) {
        _flightPeriod.addSegmentCabin (
_flightPeriod._itSegment,
                                       _flightPeriod.
_itSegmentCabin);
      } else {
        _flightPeriod.addSegmentCabin (
_flightPeriod._itSegmentCabin);
      }
    }

    // //////////////////////////////////////////////////////////////////
    storeFamilyCode::
    storeFamilyCode (FlightPeriodStruct&
ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeFamilyCode::operator() (int iCode)
 const {
      std::ostringstream ostr;
      ostr << iCode;
      _flightPeriod._itSegmentCabin._itFamilyCode
 = ostr.str();
    }

    // //////////////////////////////////////////////////////////////////
    storeFClasses::
    storeFClasses (FlightPeriodStruct&
ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    void storeFClasses::operator() (iterator_t
 iStr,
                                     iterator_t iStrEnd) const {
      std::string lClasses (iStr, iStrEnd);
      FareFamilyStruct lFareFamily(_flightPeriod.
_itSegmentCabin._itFamilyCode,
                                     lClasses);

      // The list of classes is the last (according to the arrival order
      // within the schedule input file) detail of the segment cabin. Hence,
      // when a list of classes is parsed, it means that the full segment
      // cabin details have already been parsed as well: the segment cabin
      // can thus be added to the segment.
      if (_flightPeriod._areSegmentDefinitionsSpecific
 == true) {
        _flightPeriod.addFareFamily (_flightPeriod
._itSegment,
                                     _flightPeriod._itSegmentCabin
,
                                     lFareFamily);
      } else {
        _flightPeriod.addFareFamily (_flightPeriod
._itSegmentCabin,
                                     lFareFamily);
      }
    }

    // //////////////////////////////////////////////////////////////////
    doEndFlight::
    doEndFlight (stdair::BomRoot& ioBomRoot,
                 FlightPeriodStruct& ioFlightPeriod)
      : ParserSemanticAction (ioFlightPeriod),
        _bomRoot (ioBomRoot) {
    }

    // //////////////////////////////////////////////////////////////////
    // void doEndFlight::operator() (char iChar) const {
    void doEndFlight::operator() (iterator_t
iStr,
                                  iterator_t iStrEnd) const {

      assert (_flightPeriod._legAlreadyDefined
== true);
      _flightPeriod._legList.push_back (_flightPeriod
._itLeg);

      // The lists of legs and cabins must be reset
      _flightPeriod._legAlreadyDefined = false;
      _flightPeriod._itLeg._cabinList.clear();

      // DEBUG: Display the result
      STDAIR_LOG_DEBUG ("FlightPeriod: " << _flightPeriod.describe
());

      // Create the FlightPeriod BOM objects, and potentially the intermediary
      // objects (e.g., Inventory).
      InventoryGenerator::createFlightPeriod (_bomRoot, _flightPeriod
);
    }


    // ///////////////////////////////////////////////////////////////////
    //
    //  Utility Parsers
    //
    // ///////////////////////////////////////////////////////////////////
    int1_p_t int1_p;

    uint2_p_t uint2_p;

    uint4_p_t uint4_p;

    uint1_4_p_t uint1_4_p;

    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");

    int1_p_t family_code_p;

    repeat_p_t class_code_list_p (chset_t("
A-Z").derived(), 1, 26);


    // //////////////////////////////////////////////////////////////////
    //  (Boost Spirit) Grammar Definition
    // //////////////////////////////////////////////////////////////////

    // //////////////////////////////////////////////////////////////////
    FlightPeriodParser::
    FlightPeriodParser (stdair::BomRoot& ioBomRoot,
                        FlightPeriodStruct& ioFlightPeriod)
      : _bomRoot (ioBomRoot),
        _flightPeriod (ioFlightPeriod) {
    }

    // //////////////////////////////////////////////////////////////////
    template<typename ScannerT>
    FlightPeriodParser::definition<ScannerT>::
    definition (FlightPeriodParser const& self)
 {

      flight_period_list = *(not_to_be_parsed
                             | flight_period )
        ;

      not_to_be_parsed =bsc::
        lexeme_d[bsc::comment_p("//") |bsc::comment_p("/*", "*/")
                 |bsc::eol_p];

      flight_period = flight_key
        >> +( ';' >> leg )
        >> ';' >> segment_section
        >> flight_period_end[doEndFlight (self._bomRoot, self.
_flightPeriod)]
        ;

      flight_period_end =
       bsc::ch_p(';')
        ;

      flight_key = airline_code
        >> ';' >> flight_number
        >> ';' >> date[storeDateRangeStart(self.
_flightPeriod)]
        >> ';' >> date[storeDateRangeEnd(self._flightPeriod)]
        >> ';' >> dow[storeDow(self._flightPeriod)]
        ;

      airline_code =bsc::
        lexeme_d[(airline_code_p)[storeAirlineCode
(self._flightPeriod)] ]
        ;

      flight_number =bsc::
        lexeme_d[(flight_number_p)[storeFlightNumber
(self._flightPeriod)] ]
        ;

      date =bsc::
        lexeme_d[(year_p)[bsc::assign_a(self._flightPeriod._itYear)]
                 >> '-'
                 >> (month_p)[bsc::assign_a(self._flightPeriod._itMonth)
]
                 >> '-'
                 >> (day_p)[bsc::assign_a(self._flightPeriod._itDay)]
                 ]
        ;

      dow =bsc::lexeme_d[ dow_p ]
        ;

      leg = leg_key >> ';' >> leg_details >> +( ';' >> leg_cabin_details )
        ;

      leg_key =
        (airport_p)[storeLegBoardingPoint(self.
_flightPeriod)]
        >> ';'
        >> (airport_p)[storeLegOffPoint(self._flightPeriod)]
        ;

      leg_details =
        time[storeBoardingTime(self._flightPeriod)]
        >> !(date_offset)
        >> ';'
        >> time[storeOffTime(self._flightPeriod)]
        >> !(date_offset)
        >> ';'
        >> time[storeElapsedTime(self._flightPeriod)]
        ;

      time =bsc::
        lexeme_d[(hours_p)[bsc::assign_a(self._flightPeriod._itHours)]
                 >> ':'
                 >> (minutes_p)[bsc::assign_a(self._flightPeriod._itMinutes)]
                 >> !(':'
                      >> (seconds_p)[bsc::assign_a(self._flightPeriod._itSeconds)])
                 ]
        ;

      date_offset =bsc::ch_p('/')
        >> (int1_p)[bsc::assign_a(self._flightPeriod._dateOffset)]
        ;

      leg_cabin_details = (cabin_code_p)[storeLegCabinCode(self._flightPeriod)]
        >> ';' >> (bsc::ureal_p)[storeCapacity(self._flightPeriod)]
        ;

      segment_key =
        (airport_p)[storeSegmentBoardingPoint(self._flightPeriod)]
        >> ';'
        >> (airport_p)[storeSegmentOffPoint(self._flightPeriod)]
        ;

      segment_section =
        generic_segment | specific_segment_list
        ;

      generic_segment =bsc::
        ch_p('0')[storeSegmentSpecificity(self._flightPeriod)]
        >> +(';' >> segment_cabin_details)
        ;

      specific_segment_list =bsc::
        ch_p('1')[storeSegmentSpecificity(self._flightPeriod)]
        >> +(';' >> segment_key >> full_segment_cabin_details)
        ;

      full_segment_cabin_details =
        +(';' >> segment_cabin_details)
        ;

      segment_cabin_details =
        (cabin_code_p)[storeSegmentCabinCode(self._flightPeriod)]
        >> ';' >> (class_code_list_p)[storeClasses(self._flightPeriod)]
        >> *(';' >> family_cabin_details)
        ;

      family_cabin_details =
        (family_code_p)[storeFamilyCode(self._flightPeriod)]
        >> ';'
        >> (class_code_list_p)[storeFClasses(self._flightPeriod)]
        ;

      // BOOST_SPIRIT_DEBUG_NODE (FlightPeriodParser);
      BOOST_SPIRIT_DEBUG_NODE (flight_period_list);
      BOOST_SPIRIT_DEBUG_NODE (flight_period);
      BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
      BOOST_SPIRIT_DEBUG_NODE (flight_period_end);
      BOOST_SPIRIT_DEBUG_NODE (flight_key);
      BOOST_SPIRIT_DEBUG_NODE (airline_code);
      BOOST_SPIRIT_DEBUG_NODE (flight_number);
      BOOST_SPIRIT_DEBUG_NODE (date);
      BOOST_SPIRIT_DEBUG_NODE (dow);
      BOOST_SPIRIT_DEBUG_NODE (leg);
      BOOST_SPIRIT_DEBUG_NODE (leg_key);
      BOOST_SPIRIT_DEBUG_NODE (leg_details);
      BOOST_SPIRIT_DEBUG_NODE (time);
      BOOST_SPIRIT_DEBUG_NODE (date_offset);
      BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
      BOOST_SPIRIT_DEBUG_NODE (segment_section);
      BOOST_SPIRIT_DEBUG_NODE (segment_key);
      BOOST_SPIRIT_DEBUG_NODE (generic_segment);
      BOOST_SPIRIT_DEBUG_NODE (specific_segment_list);
      BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
      BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
      BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
    }

    // //////////////////////////////////////////////////////////////////
    template<typename ScannerT>
   bsc::rule<ScannerT> const&
    FlightPeriodParser::definition<ScannerT>::start() const {
      return flight_period_list;
    }

  }


  //
  //  Entry class for the file parser
  //

  // //////////////////////////////////////////////////////////////////////
  FlightPeriodFileParser::
  FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
                          const stdair::Filename_T& iFilename)
    : _filename (iFilename), _bomRoot (ioBomRoot) {
    init();
  }

  // //////////////////////////////////////////////////////////////////////
  void FlightPeriodFileParser::init() {
    // Check that the file exists and is readable
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (_filename);

    if (doesExistAndIsReadable == false) {
      STDAIR_LOG_ERROR ("The schedule file " << _filename
                        << " does not exist or can not be read.");

      throw ScheduleInputFileNotFoundException ("The schedule file " + _filename
                                                + " does not exist or can not be read");
    }

    // Open the file
    _startIterator = iterator_t (_filename);

    // Check the filename exists and can be open
    if (!_startIterator) {
      STDAIR_LOG_ERROR ("The schedule file " << _filename << " can not be open."
                          << std::endl);

      throw ScheduleInputFileNotFoundException ("The file " + _filename
                                                + " does not exist or can not be read");
    }

    // Create an EOF iterator
    _endIterator = _startIterator.make_end();
  }

  // //////////////////////////////////////////////////////////////////////
  bool FlightPeriodFileParser::generateInventories() {
    bool oResult = false;

    STDAIR_LOG_DEBUG ("Parsing schedule input file: " << _filename);

    // Initialise the parser (grammar) with the helper/staging structure.
    ScheduleParserHelper::FlightPeriodParser lFPParser (_bomRoot,
                                                        _flightPeriod);

    // Launch the parsing of the file and, thanks to the doEndFlight
    // call-back structure, the building of the whole BomRoot BOM
    // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
    bsc::parse_info<iterator_t> info =
      bsc::parse (_startIterator, _endIterator, lFPParser,
                  bsc::space_p - bsc::eol_p);

    // Retrieves whether or not the parsing was successful
    oResult = info.hit;

    const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
    if (oResult == true) {
      STDAIR_LOG_DEBUG ("Parsing of schedule input file: " << _filename
                        << " succeeded: read " << info.length
                        << " characters. The input file has "
                        << hasBeenFullyReadStr
                        << "been fully read. Stop point: " << info.stop);

    } else {
      // TODO: decide whether to throw an exception
      STDAIR_LOG_ERROR ("Parsing of schedule input file: " << _filename
                        << " failed: read " << info.length
                        << " characters. The input file has "
                        << hasBeenFullyReadStr
                        << "been fully read. Stop point: " << info.stop);
    }

    return oResult;
  }

}
```

24.109 airsched/command/ScheduleParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp>
```

Classes

- struct AIRSCHED::ScheduleParserHelper::ParserSemanticAction
- struct AIRSCHED::ScheduleParserHelper::storeAirlineCode
- struct AIRSCHED::ScheduleParserHelper::storeFlightNumber
- struct AIRSCHED::ScheduleParserHelper::storeDateRangeStart
- struct AIRSCHED::ScheduleParserHelper::storeDateRangeEnd
- struct AIRSCHED::ScheduleParserHelper::storeDow
- struct AIRSCHED::ScheduleParserHelper::storeLegBoardingPoint
- struct AIRSCHED::ScheduleParserHelper::storeLegOffPoint
- struct AIRSCHED::ScheduleParserHelper::storeBoardingTime
- struct AIRSCHED::ScheduleParserHelper::storeOffTime
- struct AIRSCHED::ScheduleParserHelper::storeElapsedTime
- struct AIRSCHED::ScheduleParserHelper::storeLegCabinCode
- struct AIRSCHED::ScheduleParserHelper::storeCapacity
- struct AIRSCHED::ScheduleParserHelper::storeSegmentSpecificity
- struct AIRSCHED::ScheduleParserHelper::storeSegmentBoardingPoint
- struct AIRSCHED::ScheduleParserHelper::storeSegmentOffPoint
- struct AIRSCHED::ScheduleParserHelper::storeSegmentCabinCode
- struct AIRSCHED::ScheduleParserHelper::storeClasses
- struct AIRSCHED::ScheduleParserHelper::storeFamilyCode
- struct AIRSCHED::ScheduleParserHelper::storeFClasses
- struct AIRSCHED::ScheduleParserHelper::doEndFlight
- struct AIRSCHED::ScheduleParserHelper::FlightPeriodParser

- struct AIRSCHED::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >
- class AIRSCHED::FlightPeriodFileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRSCHED
- namespace AIRSCHED::ScheduleParserHelper

24.110 ScheduleParserHelper.hpp

```
#ifndef __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP
#define __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/command/CmdAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>
#include <airsched/basic/BasParserTypes.hpp>
#include <airsched/bom/FlightPeriodStruct.hpp
>

// Forward declarations
namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  namespace ScheduleParserHelper {

    // ///////////////////////////////////////////////////////////////////
    //  Semantic actions
    // ///////////////////////////////////////////////////////////////////
    struct ParserSemanticAction {
      ParserSemanticAction (FlightPeriodStruct
&);
      FlightPeriodStruct& _flightPeriod;
    };

    struct storeAirlineCode : public ParserSemanticAction
 {
      storeAirlineCode (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeFlightNumber : public ParserSemanticAction
 {
      storeFlightNumber (FlightPeriodStruct&
);
      void operator() (unsigned int iNumber) const;
    };

    struct storeDateRangeStart : public ParserSemanticAction
 {
      storeDateRangeStart (FlightPeriodStruct
&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeDateRangeEnd : public ParserSemanticAction
 {
      storeDateRangeEnd (FlightPeriodStruct&
);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeDow : public ParserSemanticAction {
      storeDow (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
    };

    struct storeLegBoardingPoint : public
  ParserSemanticAction {
      storeLegBoardingPoint (FlightPeriodStruct
  &);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeLegOffPoint : public ParserSemanticAction
   {
      storeLegOffPoint (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeBoardingTime : public ParserSemanticAction
   {
      storeBoardingTime (FlightPeriodStruct&
  );
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeOffTime : public ParserSemanticAction
   {
      storeOffTime (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeElapsedTime : public ParserSemanticAction
   {
      storeElapsedTime (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeLegCabinCode : public ParserSemanticAction
   {
      storeLegCabinCode (FlightPeriodStruct&
  );
      void operator() (char iChar) const;
    };

    struct storeCapacity : public ParserSemanticAction
   {
      storeCapacity (FlightPeriodStruct&);
      void operator() (double iReal) const;
    };

    struct storeSegmentSpecificity : public
  ParserSemanticAction {
      storeSegmentSpecificity (FlightPeriodStruct
  &);
      void operator() (char iChar) const;
    };

    struct storeSegmentBoardingPoint : public
  ParserSemanticAction {
      storeSegmentBoardingPoint (FlightPeriodStruct
  &);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeSegmentOffPoint : public
  ParserSemanticAction {
      storeSegmentOffPoint (FlightPeriodStruct
  &);
      void operator() (iterator_t iStr, iterator_t
   iStrEnd) const;
    };

    struct storeSegmentCabinCode : public
  ParserSemanticAction {
      storeSegmentCabinCode (FlightPeriodStruct
  &);
      void operator() (char iChar) const;
    };

    struct storeClasses : public ParserSemanticAction
   {
      storeClasses (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t
 iStrEnd) const;
    };

    struct storeFamilyCode : public ParserSemanticAction {
      storeFamilyCode (FlightPeriodStruct&);
      void operator() (int iCode) const;
    };

    struct storeFClasses : public ParserSemanticAction {
      storeFClasses (FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t iStrEnd) const;
    };

    struct doEndFlight : public ParserSemanticAction {
      doEndFlight (stdair::BomRoot&, FlightPeriodStruct&);
      void operator() (iterator_t iStr, iterator_t iStrEnd) const;
      stdair::BomRoot& _bomRoot;
    };


    //
    //  (Boost Spirit) Grammar Definition
    //

    struct FlightPeriodParser :
      public boost::spirit::classic::grammar<FlightPeriodParser> {

      FlightPeriodParser (stdair::BomRoot&, FlightPeriodStruct&);

      template <typename ScannerT>
      struct definition {
        definition (FlightPeriodParser const& self);

        // Instantiation of rules
        boost::spirit::classic::rule<ScannerT> flight_period_list, flight_period,
          not_to_be_parsed, flight_period_end, flight_key, airline_code,
          flight_number, date, dow, time, date_offset,
          leg, leg_key, leg_details, leg_cabin_details,
          segment_section, segment_key, full_segment_cabin_details,
          segment_cabin_details, full_family_cabin_details,
          family_cabin_details, generic_segment, specific_segment_list;

        boost::spirit::classic::rule<ScannerT> const& start() const;
      };

      // Parser Context
      stdair::BomRoot& _bomRoot;
      FlightPeriodStruct& _flightPeriod;
    };

  }

  //
  //  Entry class for the file parser
  //

  class FlightPeriodFileParser : public stdair::CmdAbstract {
  public:
    FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
                            const stdair::Filename_T& iFilename);

    bool generateInventories ();

  private:
    void init();

  private:
    // Attributes
    stdair::Filename_T _filename;

    iterator_t _startIterator;
    iterator_t _endIterator;

    stdair::BomRoot& _bomRoot;

    FlightPeriodStruct _flightPeriod;
  };

}
#endif // __AIRSCHED_CMD_SCHEDULEPARSERHELPER_HPP
```

24.111 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

24.112 SegmentPathGenerator.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <vector>
// StdAir
#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>
// AirSched
#include <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/OriginDestinationSet.hpp
>
#include <airsched/bom/SegmentPathPeriod.hpp>
#include <airsched/command/SegmentPathGenerator.hpp
>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  createSegmentPathNetwork (const stdair::BomRoot&
iBomRoot) {

    // Build the list of single-segment segment path objects.
    const stdair::InventoryList_T& lInventoryList =
      stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
    for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin()
;
         itInv != lInventoryList.end(); ++itInv) {
```

```
      const stdair::Inventory* lCurrentInventory_ptr = *itInv;
      assert (lCurrentInventory_ptr != NULL);

      //
      createSinglePaths (*lCurrentInventory_ptr);
    }

    // Build the list of i-fixed-length segment path objects. In other words,
    // build the whole segment path network.
    for (stdair::NbOfSegments_T i = 2;
         i <= stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND; ++i) {
      buildSegmentPathNetwork (iBomRoot, i);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  createSinglePaths (const stdair::Inventory& iInventory) {

    const stdair::FlightPeriodList_T& lFlightPeriodList =
      stdair::BomManager::getList<stdair::FlightPeriod> (iInventory);
    for (stdair::FlightPeriodList_T::const_iterator itFlightPeriod =
           lFlightPeriodList.begin();
         itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
      const stdair::FlightPeriod* lCurrentFlightPeriod_ptr = *itFlightPeriod;
      assert (lCurrentFlightPeriod_ptr != NULL);

      //
      createSinglePaths (*lCurrentFlightPeriod_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  createSinglePaths (const stdair::FlightPeriod& iFlightPeriod) {

    const stdair::SegmentPeriodList_T& lSegmentPeriodList =
      stdair::BomManager::getList<stdair::SegmentPeriod> (iFlightPeriod);
    for (stdair::SegmentPeriodList_T::const_iterator itSegmentPeriod =
           lSegmentPeriodList.begin();
         itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
      stdair::SegmentPeriod* lCurrentSegmentPeriod_ptr = *itSegmentPeriod;
      assert (lCurrentSegmentPeriod_ptr != NULL);

      //
      createSinglePath (*lCurrentSegmentPeriod_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  createSinglePath (stdair::SegmentPeriod& ioSegmentPeriod) {

    // Retrieve the BOM tree root
    const stdair::AirportCode_T& lOrigin = ioSegmentPeriod.getBoardingPoint();
    const stdair::FlightPeriod& lFlightPeriod =
      stdair::BomManager::getParent<stdair::FlightPeriod> (ioSegmentPeriod);
    const stdair::Inventory& lInventory =
      stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
    stdair::BomRoot& lBomRoot =
      stdair::BomManager::getParent<stdair::BomRoot> (lInventory);

    // Retrieve the ReachableUniverse (if existing) which corresponds
    // to the origin. If it does not exist, then create one.
    ReachableUniverse* lReachableUniverse_ptr =
      stdair::BomManager::getObjectPtr<ReachableUniverse> (lBomRoot, lOrigin);
    if (lReachableUniverse_ptr == NULL) {
      ReachableUniverseKey lKey (lOrigin);
      lReachableUniverse_ptr =
        &stdair::FacBom<ReachableUniverse>::instance().create (lKey);
      stdair::FacBomManager::addToListAndMap (lBomRoot, *lReachableUniverse_ptr
);
      stdair::FacBomManager::linkWithParent (lBomRoot, *lReachableUniverse_ptr)
;
    }
    assert (lReachableUniverse_ptr != NULL);

    //
    createSinglePath (*lReachableUniverse_ptr, ioSegmentPeriod);
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  createSinglePath (ReachableUniverse& ioReachableUniverse,
                    stdair::SegmentPeriod& ioSegmentPeriod) {

    const stdair::AirportCode_T& lDestination = ioSegmentPeriod.getOffPoint();

    // Retrieve the origin-destination set (if existing) which corresponds
    // to the destination. If it does not exist, then create one.
    OriginDestinationSet* lOriginDestinationSet_ptr =
      stdair::BomManager::getObjectPtr<OriginDestinationSet>(
ioReachableUniverse,
                                                             lDestination);
    if (lOriginDestinationSet_ptr == NULL) {
      OriginDestinationSetKey lKey (lDestination);

      lOriginDestinationSet_ptr =
        &stdair::FacBom<OriginDestinationSet>::instance().create (lKey);
      stdair::FacBomManager::addToListAndMap (ioReachableUniverse,
                                              *lOriginDestinationSet_ptr);
      stdair::FacBomManager::linkWithParent (ioReachableUniverse,
                                             *lOriginDestinationSet_ptr);
    }
    assert (lOriginDestinationSet_ptr != NULL);

    // Create a segment path period and add it to the corresponding
    // origin-destination set and reachable-universe.
    const stdair::FlightPeriod& lFlightPeriod =
      stdair::BomManager::getParent<stdair::FlightPeriod> (ioSegmentPeriod);
    const stdair::PeriodStruct& lPeriodOfFlight = lFlightPeriod.getPeriod();

    // The departure period of the segment is the departure period of
    // the flight plus the boarding date offset of the segment.
    const stdair::DateOffset_T& lBoardingDateOffset =
      ioSegmentPeriod.getBoardingDateOffset();

    const stdair::PeriodStruct lPeriodOfSegment =
      lPeriodOfFlight.addDateOffset (lBoardingDateOffset);

    const stdair::Duration_T& lBoardingTime = ioSegmentPeriod.getBoardingTime()
;
    const stdair::Duration_T& lElapsed = ioSegmentPeriod.getElapsedTime();

    DateOffsetList_T lDateOffsetList;
    const stdair::DateOffset_T lFirstDateOffset (0);
    lDateOffsetList.push_back (lFirstDateOffset);

    const SegmentPathPeriodKey lSegmentPathKey (lPeriodOfSegment,
                                                lBoardingTime, lElapsed,
                                                lDateOffsetList, 1);

    SegmentPathPeriod& lSegmentPathPeriod =
      stdair::FacBom<SegmentPathPeriod>::instance().create (lSegmentPathKey);

    addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod);

    // Link the SegmentPathPeriod object with its parent, namely
    // OriginDestinationSet
    stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
                                      lSegmentPathPeriod);
    stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
                                           lSegmentPathPeriod);

    // Link the SegmentPathPeriod and SegmentPeriod objects. Note that
    // the SegmentPeriod object has already a parent, namely FlightPeriod.
    stdair::FacBomManager::addToList (lSegmentPathPeriod,
                                      ioSegmentPeriod);
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  addSegmentPathPeriod (ReachableUniverse& ioReachableUniverse,
                        const SegmentPathPeriod& iSegmentPathPeriod) {

    const stdair::NbOfSegments_T& lNbOfSegments =
      iSegmentPathPeriod.getNbOfSegments();

    assert (lNbOfSegments > 0
            && lNbOfSegments <= stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND);

    // If needed, initialise the list of lists with empty fixed-length
    // segment path period lists.

    SegmentPathPeriodListList_T&
lSegmentPathPeriodListList =
      ioReachableUniverse._segmentPathPeriodListList;
    while (lSegmentPathPeriodListList.size() < lNbOfSegments) {
      SegmentPathPeriodLightList_T
lSegmentPathPeriodList;
      lSegmentPathPeriodListList.push_back (lSegmentPathPeriodList);
    }

    // Retrieve the i-fixed-length segment path period list (i = number of
    // segments).
    SegmentPathPeriodLightList_T&
lSegmentPathPeriodList =
      lSegmentPathPeriodListList.at (lNbOfSegments-1);

    // Add the SegmentPathPeriod to that fixed-length-path list.
    lSegmentPathPeriodList.push_back (&iSegmentPathPeriod);
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  buildSegmentPathNetwork (const stdair::BomRoot& iBomRoot,
                           const stdair::NbOfSegments_T& lNbOfSegments) {

    const ReachableUniverseList_T&
lReachableUniverseList =
      stdair::BomManager::getList<ReachableUniverse> (iBomRoot);
    for (ReachableUniverseList_T::const_iterator itReachableUniverse =
           lReachableUniverseList.begin();
         itReachableUniverse != lReachableUniverseList.end();
         ++itReachableUniverse) {
      ReachableUniverse* lReachableUniverse_ptr = *itReachableUniverse;
      assert (lReachableUniverse_ptr != NULL);

      //
      buildSegmentPathNetwork (*lReachableUniverse_ptr, lNbOfSegments);
    }
  }

  // //////////////////////////////////////////////////////////////////////
  void SegmentPathGenerator::
  buildSegmentPathNetwork (ReachableUniverse& ioReachableUniverse,
                           const stdair::NbOfSegments_T& iNbOfSegments) {

    // The goal of that method is to build the i-fixed-length
    // segment path period objects, knowing that all the
    // lower-fixed-length segment path period objects have already been
    // built during the previous steps. Once an i-fixed-length
    // segment path period object is created, it is added to the list of
    // the (fixed-length segment path period object) lists.

    // Hence, at that iteration, by construction, the list of the
    // (fixed-length segment path period object) lists should already get
    // a size of i-1, if there were such possibilities (in terms of
    // segment path period). In that case, at the end of the method, its
    // size should be of i.

    // If the size of the list of the (fixed-length segment path period
    // object) lists is (strictly) less than i-1, it means that that
    // reachable universe has no more possibilities of destinations. We
    // are thus done at that stage.
    const SegmentPathPeriodListList_T&
lSegmentPathPeriodListList =
      ioReachableUniverse.getSegmentPathPeriodListList();
    const unsigned short lNbOfSegments_m1 = iNbOfSegments - 1;
    assert (lNbOfSegments_m1 >= 0);
    if (lSegmentPathPeriodListList.size() < lNbOfSegments_m1) {
      return;
    }

    // Retrieve the (i-1)-fixed-length segment path period list (i = number of
    // segments).

    // Note that a STL vector starts at 0, whereas the number of segments
    // starts at 1. Hence, (i-1) for the length (in number of segments)
    // corresponds to [iNbOfSegments-2] for the STL vector.

    // As the lSegmentPathPeriodListList may change during the next loop
    // iterations (as some SegmentPathPeriod objects are created and linked to
    // ReachableUniverse), we need to take the initial copy of that list.
    const SegmentPathPeriodLightList_T
lSegmentPathPeriodLightList_im1 =
      lSegmentPathPeriodListList.at (iNbOfSegments-2);

    // Iterate on the (i-1)-fixed-length segment path period objects, in order
    // to build a i-fixed-length segment path period objects.
    // There are two steps:
    // 1. Retrieve the airport-dates at a (i-1) length (in number of segments)
    //    of the origin airport-date.
    // 2. From each of such (i-1) airport-date, add the single-segment pathes
    //    to the (i-1)-length pathes, so as to make i-length pathes.
    for (SegmentPathPeriodLightList_T::const_iterator itSegmentPathPeriodList =
           lSegmentPathPeriodLightList_im1.begin();
         itSegmentPathPeriodList != lSegmentPathPeriodLightList_im1.end();
         ++itSegmentPathPeriodList) {
      const SegmentPathPeriod* lSegmentPathPeriod_im1_ptr =
        *itSegmentPathPeriodList;
      assert (lSegmentPathPeriod_im1_ptr != NULL);

      // Get the reachable-universe departing from the destination of
      // the current segment path period.
      const stdair::AirportCode_T& lDestination_im1 =
        lSegmentPathPeriod_im1_ptr->getDestination();
      const stdair::BomRoot& lBomRoot =
        stdair::BomManager::getParent<stdair::BomRoot> (ioReachableUniverse);
      const ReachableUniverse* lReachableUniverseFromDestination_im1_ptr =
        stdair::BomManager::getObjectPtr<ReachableUniverse> (lBomRoot,
                                                             lDestination_im1);

      // If there is no ReachableUniverse corresponding to the destination (off
      // point of the last SegmentDate), it means that the destination is
      // an end point (no other SegmentDate is starting from there).
      // Hence, there is nothing else to do for now for that (final)
      // destination, and we can process the next (i-1)-segment path period.
      if (lReachableUniverseFromDestination_im1_ptr == NULL) {
        continue;
      }
      assert (lReachableUniverseFromDestination_im1_ptr != NULL);

      // Retrieve the single-segment segment path period list,
      // so as to make a i-length SegmentPathPeriod.
      const SegmentPathPeriodListList_T&
        lSegmentPathPeriodListListFromDestination_im1 =
        lReachableUniverseFromDestination_im1_ptr->
        getSegmentPathPeriodListList();
      assert (lSegmentPathPeriodListListFromDestination_im1.size() >= 1);

      // As the lSegmentPathPeriodListListFromDestination_im1 may change during
      // the next loop iterations (as some SegmentPathPeriod objects are
      // created and linked to ReachableUniverse), we need to take the initial
      // copy of that list.
      const SegmentPathPeriodLightList_T
lSingleSegmentPathPeriodLightListFromDestination_im1 =
        lSegmentPathPeriodListListFromDestination_im1.at (0);

      for (SegmentPathPeriodLightList_T::const_iterator
             itSegmentPathPeriodFromDestination_im1 =
             lSingleSegmentPathPeriodLightListFromDestination_im1.begin();
           itSegmentPathPeriodFromDestination_im1
             != lSingleSegmentPathPeriodLightListFromDestination_im1.end();
           ++itSegmentPathPeriodFromDestination_im1) {
        const SegmentPathPeriod*
lSingleSegmentPathPeriodFromDestination_im1_ptr=
          *itSegmentPathPeriodFromDestination_im1;
        assert (lSingleSegmentPathPeriodFromDestination_im1_ptr != NULL);

        // Check if the (i-1)-length segment path period can be fused with the
        // single segment 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.
        SegmentPathPeriodKey lSegmentPathPeriodKey_i =
          lSegmentPathPeriod_im1_ptr->connectWithAnotherSegment (*
lSingleSegmentPathPeriodFromDestination_im1_ptr);

        if (lSegmentPathPeriodKey_i.isValid () == false) {
          continue;
        }

        // Get the off point of the single-segment SegmentPathPeriod
        // attached to the intermediate destination (im1). That off point is
        // at a length i of the initial ReachableUniverse: (i-1) + 1.
        const stdair::AirportCode_T& lDestination_i =
          lSingleSegmentPathPeriodFromDestination_im1_ptr->getDestination();

        // Build the i-length SegmentPathPeriod
        // Get the parameters of the last segment
        stdair::SegmentPeriod* lSegmentPeriod_1_ptr =
          lSingleSegmentPathPeriodFromDestination_im1_ptr->
getFirstSegmentPeriod();
        assert (lSegmentPeriod_1_ptr != NULL);

        // Calculate the number of airlines flown by the i-length
        // segment path period
        const stdair::FlightPeriod& lFlightPeriod = stdair::BomManager::
          getParent<stdair::FlightPeriod> (*lSegmentPeriod_1_ptr);
        const stdair::Inventory& lInventory =
          stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
        const stdair::AirlineCode_T& lAirlineCode_1 =lInventory.getAirlineCode(
);
        stdair::NbOfAirlines_T lNbOfAirlines_i =
          lSegmentPathPeriod_im1_ptr->getNbOfAirlines();
        if (lSegmentPathPeriod_im1_ptr->isAirlineFlown(lAirlineCode_1) == false
){
          ++lNbOfAirlines_i;
        }
        lSegmentPathPeriodKey_i.setNbOfAirlines (lNbOfAirlines_i);

        // Create the new segment path and add it to the dedicated lists.
        OriginDestinationSet* lOriginDestinationSet_ptr = stdair::BomManager::
          getObjectPtr<OriginDestinationSet>(ioReachableUniverse,lDestination_i
);
        if (lOriginDestinationSet_ptr == NULL) {
          OriginDestinationSetKey lKey (lDestination_i);
          lOriginDestinationSet_ptr =
            &stdair::FacBom<OriginDestinationSet>::instance().create (lKey);
          stdair::FacBomManager::addToListAndMap (ioReachableUniverse,
                                                  *lOriginDestinationSet_ptr);
          stdair::FacBomManager::linkWithParent (ioReachableUniverse,
                                                 *lOriginDestinationSet_ptr);
        }
        assert (lOriginDestinationSet_ptr != NULL);


        SegmentPathPeriod& lSegmentPathPeriod_i = stdair::
          FacBom<SegmentPathPeriod>::instance().create (lSegmentPathPeriodKey_i
);
        stdair::FacBomManager::addToList (*lOriginDestinationSet_ptr,
                                             lSegmentPathPeriod_i);
        stdair::FacBomManager::linkWithParent (*lOriginDestinationSet_ptr,
                                               lSegmentPathPeriod_i);

        // Clone the list of SegmentPeriod references of the given
        // SegmentPathPeriod object (passed as the second parameter).
        stdair::FacBomManager::
          cloneHolder<stdair::SegmentPeriod> (lSegmentPathPeriod_i,
                                              *lSegmentPathPeriod_im1_ptr);


        // Add the SegmentPeriod reference to the dedicated list within
        // the SegmentPathPeriod. Note that this must be done before
        // the link between the SegmentPathPeriod and
        // ReachableUniverse, as that latter method uses the number of
        // segments within the SegmentPathPeriod object.
        stdair::FacBomManager::addToList (lSegmentPathPeriod_i,
                                          *lSegmentPeriod_1_ptr);

        addSegmentPathPeriod (ioReachableUniverse, lSegmentPathPeriod_i);
      }
    }
  }
}
```

24.113 airsched/command/SegmentPathGenerator.hpp File Reference

```
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airsched/AIRSCHED_Types.hpp>
```

Classes

- class AIRSCHED::SegmentPathGenerator

  Class handling the generation / instantiation of the network BOM.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRSCHED

24.114 SegmentPathGenerator.hpp

```
#ifndef __AIRSCHED_CMD_SEGMENTPATHGENERATOR_HPP
#define __AIRSCHED_CMD_SEGMENTPATHGENERATOR_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>

namespace stdair {
  class BomRoot;
  class Inventory;
  class FlightPeriod;
  class SegmentPeriod;
}

namespace AIRSCHED {

  class ReachableUniverse;
  class OriginDestinationSet;
  class SegmentPathPeriod;


  class SegmentPathGenerator : public stdair::CmdAbstract {
  public:
    static void createSegmentPathNetwork (const
stdair::BomRoot&);

  private:
    static void createSinglePaths (const stdair::Inventory&);
    static void createSinglePaths (const stdair::FlightPeriod&);

    static void createSinglePath (stdair::SegmentPeriod&);
    static void createSinglePath (ReachableUniverse&,
stdair::SegmentPeriod&);

    static void buildSegmentPathNetwork (const stdair::BomRoot&,
                                         const stdair::NbOfSegments_T&);
    static void buildSegmentPathNetwork (ReachableUniverse&,
                                         const stdair::NbOfSegments_T&);

    static void addSegmentPathPeriod (ReachableUniverse&,
                                      const SegmentPathPeriod&
);
  };

}
#endif // __AIRSCHED_CMD_SEGMENTPATHGENERATOR_HPP
```

24.115 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

24.116 SegmentPathProvider.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
#include <sstream>
// StdAir
#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>
// AirSched
#include <airsched/bom/ReachableUniverse.hpp>
#include <airsched/bom/OriginDestinationSet.hpp
>
#include <airsched/bom/SegmentPathPeriod.hpp>
#include <airsched/command/SegmentPathProvider.hpp
>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathProvider::
  buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
                        const stdair::BomRoot& iBomRoot,
                        const stdair::BookingRequestStruct& iBookingRequest) {
    // Retrieve  the reachable  universe object  corresponding  to the
    // origin of the booking request.
    const stdair::AirportCode_T& lOrigin = iBookingRequest.getOrigin ();
    const ReachableUniverse* lReachableUniverse_ptr =
      stdair::BomManager::getObjectPtr<ReachableUniverse> (iBomRoot, lOrigin);
    if (lReachableUniverse_ptr != NULL) {
      buildSegmentPathList (ioTravelSolutionList, *lReachableUniverse_ptr,
                            iBookingRequest);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathProvider::
  buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
                        const ReachableUniverse& iReachableUniverse,
                        const stdair::BookingRequestStruct& iBookingRequest) {
    // Retrieve the origin-destination set objet correponding to the
    // destination of the booking request.
    const stdair::AirportCode_T& lDestination = iBookingRequest.getDestination(
);
    const OriginDestinationSet* lOriginDestinationSet_ptr =
      stdair::BomManager::getObjectPtr<OriginDestinationSet> (
iReachableUniverse,
                                                              lDestination);
    if (lOriginDestinationSet_ptr != NULL) {
      buildSegmentPathList (ioTravelSolutionList, *lOriginDestinationSet_ptr,
                            iBookingRequest);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathProvider::
  buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
                        const OriginDestinationSet& iOriginDestinationSet,
                        const stdair::BookingRequestStruct& iBookingRequest) {
    // Retrieve the departure date of the booking request.
    const stdair::Date_T& lPreferedDepartureDate =
      iBookingRequest.getPreferedDepartureDate ();

    // Browse the list of segment path periods and find those which content
    // the prefered departure date.
    const SegmentPathPeriodList_T&
lSegmentPathPeriodList =
      stdair::BomManager::getList<SegmentPathPeriod> (iOriginDestinationSet);
    for (SegmentPathPeriodList_T::const_iterator itSegmentPath =
           lSegmentPathPeriodList.begin ();
         itSegmentPath != lSegmentPathPeriodList.end (); ++itSegmentPath) {
      const SegmentPathPeriod* lCurrentSegmentPath_ptr = *itSegmentPath;
      assert (lCurrentSegmentPath_ptr != NULL);
      if (lCurrentSegmentPath_ptr->isDepartureDateValid(lPreferedDepartureDate)){
        buildSegmentPathList (ioTravelSolutionList, *lCurrentSegmentPath_ptr,
                              iBookingRequest);
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPathProvider::
  buildSegmentPathList (stdair::TravelSolutionList_T& ioTravelSolutionList,
                        const SegmentPathPeriod& iSegmentPathPeriod,
                        const stdair::BookingRequestStruct& iBookingRequest) {
    // Create a new travel solution.
    stdair::TravelSolutionStruct lTravelSolution;

    // Browse the list of segments and retrieve the necessary informations
    // for identifying the corresponding segment-date.
    const stdair::Date_T& lPreferedDepartureDate =
      iBookingRequest.getPreferedDepartureDate ();
    const stdair::SegmentPeriodList_T& lSegmentPeriodList =
      stdair::BomManager::getList<stdair::SegmentPeriod> (iSegmentPathPeriod);
    const DateOffsetList_T& lBoardingDateOffsetList =
      iSegmentPathPeriod.getBoardingDateOffsetList ();
    assert (lSegmentPeriodList.size() == lBoardingDateOffsetList.size());
    DateOffsetList_T::const_iterator itOffset = lBoardingDateOffsetList.begin();
    for (stdair::SegmentPeriodList_T::const_iterator itSegment =
           lSegmentPeriodList.begin();
         itSegment != lSegmentPeriodList.end(); ++itSegment) {
      const stdair::SegmentPeriod* lSegmentPeriod_ptr = *itSegment;
      assert (lSegmentPeriod_ptr != NULL);
      const stdair::DateOffset_T& lBoardingDateOffset = *itOffset;

      // Find the corresponding segment-date within the segment period.
      const stdair::DateOffset_T& lSegmentBoardingDateOffset =
        lSegmentPeriod_ptr->getBoardingDateOffset();
      const stdair::Date_T& lReferenceFlightDate = lPreferedDepartureDate
        + lBoardingDateOffset - lSegmentBoardingDateOffset;

      // Build the whole segment-date key string.
      const stdair::FlightPeriod& lFlightPeriod =
        stdair::BomManager::getParent<stdair::FlightPeriod> (*lSegmentPeriod_ptr);
      const stdair::Inventory& lInventory =
        stdair::BomManager::getParent<stdair::Inventory> (lFlightPeriod);
      const stdair::Duration_T lBoardingTime = lSegmentPeriod_ptr->getBoardingTime();
      std::ostringstream oStr;
      oStr << lInventory.getAirlineCode()
           << stdair::DEFAULT_KEY_FLD_DELIMITER
           << lFlightPeriod.getFlightNumber()
           << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
           << boost::gregorian::to_simple_string (lReferenceFlightDate)
           << stdair::DEFAULT_KEY_FLD_DELIMITER
           << lSegmentPeriod_ptr->getBoardingPoint()
           << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
           << lSegmentPeriod_ptr->getOffPoint()
           << stdair::DEFAULT_KEY_FLD_DELIMITER
           << lBoardingTime;

      lTravelSolution.addSegment (oStr.str());

      ++itOffset;
    }
    ioTravelSolutionList.push_back (lTravelSolution);
  }

}
```

24.117 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

24.118 SegmentPathProvider.hpp

```
#ifndef __AIRSCHED_COM_CMD_SEGMENTPATHPROVIDER_HPP
#define __AIRSCHED_COM_CMD_SEGMENTPATHPROVIDER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>

namespace stdair {
  class BomRoot;
  struct BookingRequestStruct;
}

namespace AIRSCHED {

  class ReachableUniverse;
  class OriginDestinationSet;
  class SegmentPathPeriod;

  class SegmentPathProvider : public stdair::CmdAbstract {
    friend class AIRSCHED_Service;

  private:
    // ///////////////// Business Methods //////////////////
    static void buildSegmentPathList (stdair::TravelSolutionList_T&,
                                      const stdair::BomRoot&,
                                      const stdair::BookingRequestStruct&);

    static void buildSegmentPathList (stdair::TravelSolutionList_T&,
                                      const ReachableUniverse&
,
                                      const stdair::BookingRequestStruct&);

    static void buildSegmentPathList (stdair::TravelSolutionList_T&,
                                      const OriginDestinationSet
&,
                                      const stdair::BookingRequestStruct&);

    static void buildSegmentPathList (stdair::TravelSolutionList_T&,
                                      const SegmentPathPeriod&
,
                                      const stdair::BookingRequestStruct&);
  };

}
#endif // __AIRSCHED_COM_CMD_SEGMENTPATHPROVIDER_HPP
```

24.119 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

24.120 Simulator.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
// AIRSCHED
#include <airsched/command/Simulator.hpp>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  void Simulator::simulate (stdair::BomRoot& ioBomRoot) {

    // Delegate to the dedicated StdAir utility class
    // std::ostringstream oStream;
    // stdair::BomManager::display (oStream, ioBomRoot);

    // DEBUG
    // STDAIR_LOG_DEBUG ("BOM Tree: ");
    // STDAIR_LOG_DEBUG (oStream.str());

    // TODO: do not hardcode the booking request (get it from the
    // demand generation module instead).
    // stdair::BookingRequestStruct ("LHR", "JFK", stdair::Date_T (2009, 01,
 16),
    //                                 stdair::DEFAULT_DATETIME, "Y", 1);
  }

}
```

24.121 airsched/command/Simulator.hpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class AIRSCHED::Simulator

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRSCHED

24.122 Simulator.hpp

```
#ifndef __AIRSCHED_COM_CMD_SIMULATOR_HPP
#define __AIRSCHED_COM_CMD_SIMULATOR_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/command/CmdAbstract.hpp>

// Forward declarations
namespace stdair {
  class BomRoot;
}

namespace AIRSCHED {

  class Simulator : public stdair::CmdAbstract {
  public:

    // ////////// Business Methods /////////
    static void simulate (stdair::BomRoot&);
  };

}
#endif // __AIRSCHED_COM_CMD_SIMULATOR_HPP
```

24.123 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

24.124 TravelSolutionParser.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <cassert>
// StdAir
#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>
// AirSched
#include <airsched/command/TravelSolutionParser.hpp
>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  bool TravelSolutionParser::
  parseInputFileAndBuildBom (const std::string&
iInputFileName) {
    bool hasReadBeenSuccessful = false;

    // Check that the file path given as input corresponds to an actual file
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (iInputFileName);
    if (doesExistAndIsReadable == false) {
      std::ostringstream oMessage;
      oMessage << "The input file, '" << iInputFileName
               << "', can not be retrieved on the file-system";
      throw stdair::FileNotFoundException (oMessage.str());
    }

    // Open the input file
    std::ifstream inputFile (iInputFileName.c_str());
    if (! inputFile) {
      STDAIR_LOG_ERROR ("Can not open input file ’" << iInputFileName << "’");
      throw new stdair::FileNotFoundException ("Can not open input file ’"
                                               + iInputFileName + "’");
    }

    char buffer[80];
    double dval = 0.0;
    std::string dvalStr;
    short i = 1;
    bool hasAllPArams = true;

    stdair::AirportCode_T dAirport;
    stdair::AirportCode_T aAirport;
    stdair::Date_T depDate;
    stdair::Duration_T depTime;
    stdair::Duration_T arTime;
    stdair::Duration_T dur;
    //bool Ref;
    stdair::AirlineCode_T airline;
    stdair::CabinCode_T cabin;
    //stdair::FlightNumber_T flightNum;
    //stdair::Fare_T fare;
    //int lagsNum;
    //bool SNS;
    //bool change;

    while (inputFile.getline (buffer, sizeof (buffer), ’;’)) {
      std::istringstream iStringStr (buffer);

      bool hasRead = false;

      if (i == 1) {
        hasAllPArams = true;
      }

      if (i>=1 && i<=14) {
        hasRead = (iStringStr >> dvalStr);
      }

      if (i == 15) {
        hasRead = (iStringStr >> dval);
      }

      if (hasRead) {
        if (i == 1) {
          dAirport = dvalStr;

        } else if (i == 2) {
          aAirport = dvalStr;
          // std::cout << "City Pair = ’" << dAiport
          // << "-" << aAirport << "’" << std::endl;

        } else if (i == 3) {
          depDate = boost::gregorian::from_simple_string (dvalStr);
          // std::cout << "Date = ’" << depDate << "’" << std::endl;

        } else if (i == 4) {
          depTime = boost::posix_time::duration_from_string (dvalStr);

        } else if (i == 5) {
          arTime = boost::posix_time::duration_from_string (dvalStr);

        } else if (i == 6) {
          dur = boost::posix_time::duration_from_string (dvalStr);

        } else if (i == 7) {
          //if (dvalStr == "refundable fare")
          //  Ref = true;
          //else Ref  = false;

        } else if (i == 8) {
          airline = dvalStr;

        } else if (i == 9) {
          cabin = dvalStr;

        } else if (i == 10) {
          //flightNum = dval;

        } else if (i == 11) {
          //fare = dval;

        } else if (i == 12) {
          //lagsNum = dval;

        } else if (i == 13) {
          //if (dvalStr == "Saturday Nigth Stay mandatory")
          //  SNS = true;
          //else SNS = false;

        } else if (i == 14) {
          //if (dvalStr == "changeable fare")
          //  change = true;
          //else change = false;
          i = 0;
        }

        //
        ++i;

      } else {
        hasAllPArams = false;
      }
    }

    if (hasAllPArams && i == 1) {
      STDAIR_LOG_DEBUG ("Successfully read");
    }

    //
    if (!inputFile.eof()) {
      STDAIR_LOG_ERROR ("Problem when reading input file '" << iInputFileName
                        << "'");
      return hasReadBeenSuccessful;
    }

    //
    hasReadBeenSuccessful = true;
    return hasReadBeenSuccessful;
  }

}
```

24.125 airsched/command/TravelSolutionParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class AIRSCHED::TravelSolutionParser

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

Namespaces

- namespace AIRSCHED

24.126 TravelSolutionParser.hpp

```
#ifndef __AIRSCHED_CMD_TRAVELSOLUTIONPARSER_HPP
#define __AIRSCHED_CMD_TRAVELSOLUTIONPARSER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>

namespace AIRSCHED {

  class TravelSolutionParser : public stdair::CmdAbstract {
  public:
    static bool parseInputFileAndBuildBom (const
stdair::Filename_T&);
  };
}
#endif // __AIRSCHED_CMD_TRAVELSOLUTIONPARSER_HPP
```

```
 */
#ifndef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_PATHS_HPP__

#define PACKAGE "airsched"
#define PACKAGE_NAME "AIRSCHED"
#define PACKAGE_VERSION "0.1.4"
#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-0.1.4"
#define MANDIR "/usr/share/man"
#define INFODIR "/usr/share/info"
#define HTMLDIR "/usr/share/doc/airsched-0.1.4/html"
#define PDFDIR "/usr/share/doc/airsched-0.1.4/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__

/*!
```

24.127 airsched-paths.hpp

```

#ifndef __AIRSCHED_PATHS_HPP__
#define __AIRSCHED_PATHS_HPP__

#define PACKAGE "airsched"
#define PACKAGE_NAME "AIRSCHED"
#define PACKAGE_VERSION "0.1.4"
#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-0.1.4"
#define MANDIR "/usr/share/man"
#define INFODIR "/usr/share/info"
#define HTMLDIR "/usr/share/doc/airsched-0.1.4/html"
#define PDFDIR "/usr/share/doc/airsched-0.1.4/html"
#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

#endif // __AIRSCHED_PATHS_HPP__

```

24.128 airsched/config/airsched-paths.hpp.in File Reference

24.129 airsched-paths.hpp.in

```

#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__

```

24.130 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

24.131 FacAIRSCHEDServiceContext.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/service/FacSupervisor.hpp>
// AirSched
#include <airsched/factory/FacAIRSCHEDServiceContext.hpp
>
#include <airsched/service/AIRSCHED_ServiceContext.hpp
>

namespace AIRSCHED {

  FacAIRSCHEDServiceContext* FacAIRSCHEDServiceContext::_instance = NULL;

  // //////////////////////////////////////////////////////////////////////
  FacAIRSCHEDServiceContext::~FacAIRSCHEDServiceContext
() {
    _instance = NULL;
  }

  // //////////////////////////////////////////////////////////////////////
  FacAIRSCHEDServiceContext&
FacAIRSCHEDServiceContext::instance () {

    if (_instance == NULL) {
      _instance = new FacAIRSCHEDServiceContext();
      assert (_instance != NULL);

      stdair::FacSupervisor::instance().
registerServiceFactory (_instance);
    }
    return *_instance;
  }

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_ServiceContext&
FacAIRSCHEDServiceContext::create () {
    AIRSCHED_ServiceContext* aServiceContext_ptr = NULL;

    aServiceContext_ptr = new AIRSCHED_ServiceContext();
    assert (aServiceContext_ptr != NULL);

    // The new object is added to the Bom pool
    _pool.push_back (aServiceContext_ptr);

    return *aServiceContext_ptr;
  }

}
```

24.132 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

24.133 FacAIRSCHEDServiceContext.hpp

```
#ifndef __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_HPP
#define __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

namespace AIRSCHED {

  class AIRSCHED_ServiceContext;

  class FacAIRSCHEDServiceContext : public
stdair::FacServiceAbstract {
  public:

    static FacAIRSCHEDServiceContext& instance
```

```
      ();

    ~FacAIRSCHEDServiceContext();

    AIRSCHED_ServiceContext& create();


  protected:
    FacAIRSCHEDServiceContext() {}

  private:
    static FacAIRSCHEDServiceContext* _instance;

  };
}
#endif // __AIRSCHED_FAC_FACAIRSCHEDSERVICECONTEXT_HPP
```

24.134 airsched/factory/FacServiceAbstract.cpp File Reference

```
#include <assert.h>
#include <airsched/service/ServiceAbstract.hpp>
#include <airsched/factory/FacServiceAbstract.hpp>
```

Namespaces

- namespace AIRSCHED

24.135 FacServiceAbstract.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// C
#include <assert.h>
// TRAVEL-CCM
#include <airsched/service/ServiceAbstract.hpp
>
#include <airsched/factory/FacServiceAbstract.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  FacServiceAbstract::~FacServiceAbstract
      () {
    clean ();
  }

  // //////////////////////////////////////////////////////////////////////
  void FacServiceAbstract::clean() {
    for (ServicePool_T::iterator itService = _pool.begin();
         itService != _pool.end(); itService++) {
      ServiceAbstract* currentService_ptr = *itService;
      assert (currentService_ptr != NULL);

      delete (currentService_ptr); currentService_ptr = NULL;
    }

    // Empty the pool of Service Factories
    _pool.clear();
  }

}
```

24.136 airsched/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class AIRSCHED::FacServiceAbstract

Namespaces

- namespace AIRSCHED

24.137 FacServiceAbstract.hpp

```
#ifndef __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP
#define __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <vector>

namespace AIRSCHED {

  // Forward declarations
  class ServiceAbstract;

  class FacServiceAbstract {
  public:

    typedef std::vector<ServiceAbstract*> ServicePool_T;

    virtual ~FacServiceAbstract();

    void clean();

  protected:
    FacServiceAbstract() {}

    ServicePool_T _pool;
  };
}
#endif // __AIRSCHED_FAC_FACSERVICEABSTRACT_HPP
```

24.138 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

24.139 AIRSCHED_Service.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#include <boost/make_shared.hpp>
// StdAir
#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>
// AirSched
#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>

namespace AIRSCHED {

  // ////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::AIRSCHED_Service() : _airschedServiceContext (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::AIRSCHED_Service (const AIRSCHED_Service& iService)
    : _airschedServiceContext (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::AIRSCHED_Service (const stdair::BasLogParams& iLogParams)
    : _airschedServiceContext (NULL) {

    // Initialise the STDAIR service handler
    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      initStdAirService (iLogParams);

    // Initialise the service context
    initServiceContext();

    // Add the StdAir service context to the AirSched service context
    // \note AirSched owns the STDAIR service resources here.
    const bool ownStdairService = true;
    addStdAirService (lSTDAIR_Service_ptr, ownStdairService);

    // Initialise the (remaining of the) context
    initAirschedService();
  }

  // ////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::AIRSCHED_Service (const stdair::BasLogParams& iLogParams,
                                      const stdair::BasDBParams& iDBParams)
    : _airschedServiceContext (NULL) {

    // Initialise the STDAIR service handler
    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      initStdAirService (iLogParams, iDBParams);

    // Initialise the service context
    initServiceContext();

    // Add the StdAir service context to the AirSched service context
    // \note AirSched owns the STDAIR service resources here.
    const bool ownStdairService = true;
    addStdAirService (lSTDAIR_Service_ptr, ownStdairService);

    // Initialise the (remaining of the) context
    initAirschedService();
  }

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::
  AIRSCHED_Service (stdair::STDAIR_ServicePtr_T ioSTDAIRServicePtr)
    : _airschedServiceContext (NULL) {

    // Initialise the service context
    initServiceContext();

    // Add the StdAir service context to the AirSched service context.
    // \note AirSched does not own the STDAIR service resources here.
    const bool doesNotOwnStdairService = false;
    addStdAirService (ioSTDAIRServicePtr, doesNotOwnStdairService);

    // Initialise the context
    initAirschedService();
  }

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_Service::~AIRSCHED_Service() {
    // Delete/Clean all the objects from memory
    finalise();
  }

  // //////////////////////////////////////////////////////////////////////
  void AIRSCHED_Service::finalise() {
    assert (_airschedServiceContext != NULL);
    // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
    _airschedServiceContext->reset();
  }

  // ////////////////////////////////////////////////////////////////////////
  void AIRSCHED_Service::initServiceContext() {
    // Initialise the service context
    AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
      FacAIRSCHEDServiceContext::instance().
create();
    _airschedServiceContext = &lAIRSCHED_ServiceContext;
  }

  // //////////////////////////////////////////////////////////////////////
  void AIRSCHED_Service::
  addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
                    const bool iOwnStdairService) {

    // Retrieve the AirSched service context
    assert (_airschedServiceContext != NULL);
    AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
      *_airschedServiceContext;

    // Store the STDAIR service object within the (AirSched) service context
    lAIRSCHED_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
                                                iOwnStdairService);
  }

  // ////////////////////////////////////////////////////////////////////////
  stdair::STDAIR_ServicePtr_T AIRSCHED_Service::
  initStdAirService (const stdair::BasLogParams& iLogParams) {

    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      boost::make_shared<stdair::STDAIR_Service> (iLogParams);

    return lSTDAIR_Service_ptr;
  }

  // ////////////////////////////////////////////////////////////////////////
  stdair::STDAIR_ServicePtr_T AIRSCHED_Service::
  initStdAirService (const stdair::BasLogParams& iLogParams,
                     const stdair::BasDBParams& iDBParams) {

    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);

    return lSTDAIR_Service_ptr;
  }

  // //////////////////////////////////////////////////////////////////////
```

```
void AIRSCHED_Service::initAirschedService() {
  // Do nothing at this stage. A sample BOM tree may be built by
  // calling the buildSampleBom() method
}

// //////////////////////////////////////////////////////////////////////
void AIRSCHED_Service::
parseAndLoad (const stdair::Filename_T& iScheduleInputFilename)
{

  // Retrieve the BOM root object.
  assert (_airschedServiceContext != NULL);
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();
  stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();

  // Parse the schedule input file, and generate the Inventories
  stdair::BasChronometer lINVGeneration; lINVGeneration.start();
  ScheduleParser::generateInventories (
iScheduleInputFilename, lBomRoot);
  const double lGenerationMeasure = lINVGeneration.elapsed();

  // DEBUG
  STDAIR_LOG_DEBUG ("Inventory generation time: " << lGenerationMeasure);
}

// //////////////////////////////////////////////////////////////////////
void AIRSCHED_Service::
parseAndLoad (const stdair::Filename_T& iScheduleInputFilename,
              const stdair::Filename_T& iODInputFilename) {

  // First, build the airline inventories from the schedule file
  parseAndLoad (iScheduleInputFilename);

  // Retrieve the BOM tree root
  assert (_airschedServiceContext != NULL);
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();
  stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();

  // Parse the O&D input file, and generate the O&D periods
  stdair::BasChronometer lOnDGeneration; lOnDGeneration.start();
  OnDParser::generateOnDPeriods (
iODInputFilename, lBomRoot);
  const double lGenerationMeasure = lOnDGeneration.elapsed();

  // DEBUG
  STDAIR_LOG_DEBUG ("O&D generation time: " << lGenerationMeasure);
}

// ///////////////////////////////////////////////////////////////////////
void AIRSCHED_Service::buildSampleBom() {

  // Retrieve the AirSched service context
  if (_airschedServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The AirSched service has "
                                                  "not been initialised");
  }
  assert (_airschedServiceContext != NULL);

  // Retrieve the AirSched service context and whether it owns the Stdair
  // service
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  const bool doesOwnStdairService =
    lAIRSCHED_ServiceContext.getOwnStdairServiceFlag();

  // Retrieve the StdAir service object from the (AirSched) service context
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();

  if (doesOwnStdairService == true) {
    //
    lSTDAIR_Service.buildSampleBom();
  }

  stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
  SegmentPathGenerator::createSegmentPathNetwork
(lBomRoot);
}

// //////////////////////////////////////////////////////////////////////
std::string AIRSCHED_Service::
```

```
jsonExport (const stdair::AirlineCode_T& iAirlineCode,
            const stdair::FlightNumber_T& iFlightNumber,
            const stdair::Date_T& iDepartureDate) const {

  // Retrieve the AirSched service context
  if (_airschedServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The AirSched service "
                                                  "has not been initialised");
  }
  assert (_airschedServiceContext != NULL);

  // Retrieve the StdAir service object from the (AirSched) service context
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();

  // Delegate the JSON export to the dedicated service
  return lSTDAIR_Service.jsonExport (iAirlineCode, iFlightNumber,
                                     iDepartureDate);
}

// //////////////////////////////////////////////////////////////////////
std::string AIRSCHED_Service::csvDisplay() const {

  // Retrieve the AirSched service context
  if (_airschedServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The AirSched service has "
                                                  "not been initialised");
  }
  assert (_airschedServiceContext != NULL);

  // Retrieve the STDAIR service object from the (AirSched) service context
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();

  // Delegate the BOM building to the dedicated service
  return lSTDAIR_Service.csvDisplay();
}

// //////////////////////////////////////////////////////////////////////
std::string AIRSCHED_Service::
csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
            const stdair::FlightNumber_T& iFlightNumber,
            const stdair::Date_T& iDepartureDate) const {

  // Retrieve the AirSched service context
  if (_airschedServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The AirSched service has "
                                                  "not been initialised");
  }
  assert (_airschedServiceContext != NULL);

  // Retrieve the STDAIR service object from the (AirSched) service context
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();

  // Delegate the BOM display to the dedicated service
  return lSTDAIR_Service.csvDisplay (iAirlineCode, iFlightNumber,
                                     iDepartureDate);
}

// //////////////////////////////////////////////////////////////////////
void AIRSCHED_Service::simulate() {

  // Retrieve the AirSched service context
  if (_airschedServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The AirSched service has "
                                                  "not been initialised");
  }
  assert (_airschedServiceContext != NULL);

  // Retrieve the BOM tree root
  AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
    *_airschedServiceContext;
  stdair::STDAIR_Service& lSTDAIR_Service =
    lAIRSCHED_ServiceContext.getSTDAIR_Service();
  stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();

  // Call the underlying Use Case (command)
  stdair::BasChronometer lSimulateChronometer; lSimulateChronometer.start();
```

```
    Simulator::simulate (lBomRoot);
    const double lSimulateMeasure = lSimulateChronometer.elapsed();

    // DEBUG
    STDAIR_LOG_DEBUG ("Simulation: " << lSimulateMeasure << " - "
                      << lAIRSCHED_ServiceContext.display());
  }

  // ////////////////////////////////////////////////////////////////////
  void AIRSCHED_Service::
  buildSegmentPathList (stdair::TravelSolutionList_T&
ioTravelSolutionList,
                        const stdair::BookingRequestStruct& iBookingRequest) {

    if (_airschedServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The AirSched service has "
                                                    "not been initialised");
    }
    assert (_airschedServiceContext != NULL);

    // Retrieve the BOM tree root
    AIRSCHED_ServiceContext& lAIRSCHED_ServiceContext =
      *_airschedServiceContext;
    stdair::STDAIR_Service& lSTDAIR_Service =
      lAIRSCHED_ServiceContext.getSTDAIR_Service();
    stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();

    // Delegate the call to the dedicated command
    stdair::BasChronometer lBuildChronometer; lBuildChronometer.start();
    SegmentPathProvider::buildSegmentPathList
 (ioTravelSolutionList,
                                              lBomRoot, iBookingRequest);
    const double lBuildMeasure = lBuildChronometer.elapsed();

    // DEBUG
    STDAIR_LOG_DEBUG ("Segment-path build: " << lBuildMeasure << " - "
                      << lAIRSCHED_ServiceContext.display());
  }

}
```

24.140 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

24.141 AIRSCHED_ServiceContext.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/STDAIR_Service.hpp>
// AirSched
#include <airsched/basic/BasConst_AIRSCHED_Service.hpp
>
#include <airsched/service/AIRSCHED_ServiceContext.hpp
>

namespace AIRSCHED {

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_ServiceContext::AIRSCHED_ServiceContext()
    : _ownStdairService (false) {
  }

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_ServiceContext::
  AIRSCHED_ServiceContext (const AIRSCHED_ServiceContext&) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  AIRSCHED_ServiceContext::~AIRSCHED_ServiceContext() {
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::STDAIR_Service& AIRSCHED_ServiceContext::getSTDAIR_Service() const {
    assert (_stdairService != NULL);
    return *_stdairService;
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string AIRSCHED_ServiceContext::shortDisplay() const {
    std::ostringstream oStr;
    oStr << "AIRSCHED_ServiceContext -- Owns StdAir service: "
         << _ownStdairService;
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string AIRSCHED_ServiceContext::display() const {
    std::ostringstream oStr;
    oStr << shortDisplay();
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string AIRSCHED_ServiceContext::describe() const {
    return shortDisplay();
  }

  // //////////////////////////////////////////////////////////////////////
  void AIRSCHED_ServiceContext::reset() {
    if (_ownStdairService == true) {
      _stdairService.reset();
    }
  }

}
```

24.142 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

24.143 AIRSCHED_ServiceContext.hpp

```
#ifndef __AIRSCHED_SVC_AIRSCHED_SERVICE_CONTEXT_HPP
#define __AIRSCHED_SVC_AIRSCHED_SERVICE_CONTEXT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost
#include <boost/shared_ptr.hpp>
// StdAir
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
// AirSched
#include <airsched/AIRSCHED_Types.hpp>

namespace AIRSCHED {

  class AIRSCHED_ServiceContext : public
stdair::ServiceAbstract {
    friend class AIRSCHED_Service;
    friend class FacAIRSCHEDServiceContext;

  private:
    // ///////////////// Getters ///////////////////
    stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
      return _stdairService;
    }

    stdair::STDAIR_Service& getSTDAIR_Service() const;

    const bool getOwnStdairServiceFlag() const {
      return _ownStdairService;
    }


  private:
    // ///////////////// Setters ///////////////////
    void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
                            const bool iOwnStdairService) {
      _stdairService = ioSTDAIR_ServicePtr;
      _ownStdairService = iOwnStdairService;
    }


  private:
    // //////////////////// Display Methods /////////////////////
    const std::string shortDisplay() const;

    const std::string display() const;

    const std::string describe() const;


  private:

    AIRSCHED_ServiceContext();

    AIRSCHED_ServiceContext (const AIRSCHED_ServiceContext&);

    void init();

    ~AIRSCHED_ServiceContext();

    void reset();


  private:
    // /////////////// Children ///////////////
    stdair::STDAIR_ServicePtr_T _stdairService;

    bool _ownStdairService;
  };

}
#endif // __AIRSCHED_SVC_AIRSCHED_SERVICE_CONTEXT_HPP
```

24.144 airsched/service/ServiceAbstract.cpp File Reference

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

Namespaces

- namespace AIRSCHED

24.145 ServiceAbstract.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// AIRSCHED
#include <airsched/service/ServiceAbstract.hpp
>

namespace AIRSCHED {

}
```

24.146 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 &i-Service)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, AIRSCHED::ServiceAbstract &ioService)

24.146.1 Function Documentation

24.146.1.1 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.

24.146.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* 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().

24.147 ServiceAbstract.hpp

```
#ifndef __AIRSCHED_SERVICEABSTRACT_HPP
#define __AIRSCHED_SERVICEABSTRACT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iostream>
#include <sstream>

namespace AIRSCHED {

  class ServiceAbstract {
  public:

    virtual ~ServiceAbstract() {}

    virtual void toStream (std::ostream& ioOut) const {}

    virtual void fromStream (std::istream& ioIn) {}

  protected:
    ServiceAbstract() {}
  };
}

template <class charT, class traits>
inline
std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream<charT, traits>& ioOut,
            const AIRSCHED::ServiceAbstract& iService)
 {
  std::basic_ostringstream<charT,traits> ostr;
  ostr.copyfmt (ioOut);
  ostr.width (0);

  // Fill string stream
  iService.toStream (ostr);

  // Print string stream
  ioOut << ostr.str();

  return ioOut;
}

template <class charT, class traits>
inline
std::basic_istream<charT, traits>&
operator>> (std::basic_istream<charT, traits>& ioIn,
            AIRSCHED::ServiceAbstract& ioService) {
  // Fill Service object with input stream
  ioService.fromStream (ioIn);
  return ioIn;
}

#endif // __AIRSCHED_SERVICEABSTRACT_HPP
```

24.148 doc/local/authors.doc File Reference

24.149 doc/local/codingrules.doc File Reference

24.150 doc/local/copyright.doc File Reference

24.151 doc/local/documentation.doc File Reference

24.152 doc/local/features.doc File Reference

24.153 doc/local/help_wanted.doc File Reference

24.154 doc/local/howto_release.doc File Reference

24.155 doc/local/index.doc File Reference

24.156 doc/local/installation.doc File Reference

24.157 doc/local/linking.doc File Reference

24.158 doc/local/test.doc File Reference

24.159 doc/local/users_guide.doc File Reference

24.160 doc/local/verification.doc File Reference

24.161 doc/tutorial/tutorial.doc File Reference

24.162 test/airsched/AirlineScheduleTestSuite.cpp File Reference

24.163 AirlineScheduleTestSuite.cpp

```

// //////////////////////////////////////////////////////////////////////
// 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/AIRSCHED_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()


```

24.164 test/airsched/AirlineScheduleTestSuite.hpp File Reference

```
#include <sstream>
#include <cppunit/extensions/HelperMacros.h>
```

Classes

- class AirlineScheduleTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (AirlineScheduleTestSuite)

24.164.1 Function Documentation

24.164.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (AirlineScheduleTestSuite)

24.165 AirlineScheduleTestSuite.hpp

```
// STL
#include <sstream>
// CPPUNIT
#include <cppunit/extensions/HelperMacros.h>

class AirlineScheduleTestSuite : public
CppUnit::TestFixture {
  CPPUNIT_TEST_SUITE (AirlineScheduleTestSuite);
//  CPPUNIT_TEST (externalMemoryManagement);
  CPPUNIT_TEST (scheduleParsing);
  CPPUNIT_TEST_SUITE_END ();
public:

  void externalMemoryManagement ();
  void scheduleParsing ();

  AirlineScheduleTestSuite ();

protected:
  std::stringstream _describeKey;
};
CPPUNIT_TEST_SUITE_REGISTRATION (
AirlineScheduleTestSuite);
```

Index