SEvMgr

1.00.2

Generated by Doxygen 1.6.1

Sun Jun 14 13:42:48 2015

Contents

1 SEvMgr Documentation

1.1 Getting Started

- Main features
- Installation
- Linking with SEvMgr
- Users Guide
- Tutorials
- Copyright and License
- Make a Difference
- Make a new release
- People

1.2 SEvMgr at SourceForge

- Project page
- Download SEvMgr
- Open a ticket for a bug or feature
- Mailing lists
- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss SEvMgr

1.3 SEvMgr Development

- Git Repository
- Coding Rules
- Documentation Rules
- Test Rules

1.4 External Libraries

- Boost (C++ STL extensions)
- Python
- MySQL client
- SOCI (C++ DB API)

1.5 Support SEvMgr

1.6 About SEvMgr

SEvMgr is a C++ library of discrete event queue management classes and functions, exclusively targeting simulation purposes. N

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

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

SEvMgr 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 SEvMgr 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 SEvMgr.

2 People

2.1 Project Admins

- Gabrielle Sabatier <`gsabatier@users.sourceforge.net`> (N)
- Denis Arnaud <`denis_arnaud@users.sourceforge.net`> (N)

2.2 Developers

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

2.3 Retired Developers

- Mehdi Ayouni <`mehdi.ayouni@gmail.com`>
- Patrick Grandjean <`pgrandjean@users.sourceforge.net`> (N)

2.4 Contributors

- Emmanuel Bastien <`ebastien@users.sourceforge.net`> (N)

2.5 Distribution Maintainers

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

Note:

(N) - `Amadeus` employees.

3 Coding Rules

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

3.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

3.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

3.3 Default Naming Rules for Classes and Structures

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

- `MyClassName`
- `MyStructName`

3.4 Default Naming Rules for Files

Files are named after the C++ class names.

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

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

3.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

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

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

4 Copyright and License

4.1 GNU LESSER GENERAL PUBLIC LICENSE

4.1.1 Version 2.1, February 1999

4.2 Preamble

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

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

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

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

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

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

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

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

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

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

For an executable, the required form of the "work that uses the Library" must include any data and utility

4.3.1 NO WARRANTY

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

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

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

```
<one line to give the library’s name and a brief idea of what it does.>
```

```
    Copyright (C) <year>  <name of author>

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

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

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

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

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

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

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

That's all there is to it!

Source

5 Documentation Rules

5.1 General Rules

All classes in SEvMgr 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 SEvMgr is shown here:

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

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

  /*!
```

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

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

private:

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

5.2 File Header

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

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

5.3 Grouping Various Parts

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

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

```
*/
```

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

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

6 Main features

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

6.1 Booking management

- Booking and cancellation requests

6.2 Revenue Management notification

- Forecast and Optimisation notification requests

6.3 Setting simulation break-points

- Simulation break-points

6.4 Other features

- CSV input file parsing
- Memory handling

7 Make a Difference

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

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

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

8 Make a new release

8.1 Introduction

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

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

8.2 Initialisation

Clone locally the full `Git project`:

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

8.3 Branch creation

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

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

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

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

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

```
vi ChangeLog
vi sevmgr.spec
```

8.4 Commit and publish the release branch

Commit the new release:

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

8.5 Update the change-log in the trunk as well

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

```
cd ~/dev/sim/sevmgrgit
git checkout trunk
vi ChangeLog
vi sevmgr.spec
```

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

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

8.6 Create distribution packages

Create the distribution packages using the following command:

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

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

8.7 Generation the RPM packages

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

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

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

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

8.8 Update distributed change log

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

8.9 Create the binary package, including the documentation

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

```
make package
```

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

8.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check `SourceForge help page on uploading software`.

8.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- `synchronise them with rsync and SSH`:

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

  where `-aiv` options mean:

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

- or use the `SourceForge Shell service`.

8.12 Make a new post

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

8.13 Send an email on the announcement mailing-list

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

9 Installation

9.1 Table of Contents

9.2 Fedora/RedHat Linux distributions

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

```
yum -y install sevmgr-devel sevmgr-doc
```

RPM packages can also be available on the `SourceForge download site`.

9.3 SEvMgr Requirements

SEvMgr 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 SEvMgr.

9.4 Basic Installation

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

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

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

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'CMakeLists.txt' is used to create the 'Makefile' files.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './cmake ..' to configure the package for your system. Running 'cmake' is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type 'make' to compile the package.
3. Optionally, type 'make check' to run any self-tests that come with the package, generally using the just-built uninstalled binaries.

4. Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
6. Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

9.5 Compilers and Options

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

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

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

See also:

Defining Variables for more details.

9.6 Compiling For Multiple Architectures

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

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

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

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

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

9.7 Installation Names

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

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

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

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

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

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of '${prefix}' at 'configure' time.

9.8 Optional Features

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

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

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

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

9.9 Particular systems

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

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

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

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

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

and if that doesn't work, try

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

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

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

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

9.10 Specifying the System Type

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

where SYSTEM can have one of these forms:

- OS
- KERNEL-OS

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

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

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

9.11 Sharing Defaults

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

9.12 Defining Variables

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

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

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

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

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

9.13 'cmake' Invocation

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

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

  See also:

  Installation Names for more details, including other options available for fine-tuning the installation locations.
- '--no-create', '-n' run the configure checks, but stop before creating any output files.

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

The 'cmake' script produces an ouput like this:

```
export LIBSUFFIX_4_CMAKE="-DLIB_SUFFIX=64"
export INSTALL_BASEDIR=/home/user/dev/deliveries
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_BASEDIR}/sevmgr-0.5.0 \
 -DWITH_STDAIR_PREFIX=${INSTALL_BASEDIR}/stdair-stable \
 -DWITH_AIRRAC_PREFIX=${INSTALL_BASEDIR}/airrac-stable \
 -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/rmol-stable \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ${LIBSUFFIX_4_CMAKE} ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: 0ee8dcc3e3dd1d1d442c4054fbfa4cacc1182e6a trunk
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   regex
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires Readline without specifying any version
-- Found Readline: /usr/include
-- Found Readline version: 6.2
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.37
-- Found StdAir version: 0.38.0
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'sevmgrlib' to CXX
-- Test 'InventoryTestSuite' to be built with 'InventoryTestSuite.cpp'
--
-- =============================================================
-- ----------------------------------
-- ---     Project Information    ---
-- ----------------------------------
-- PROJECT_NAME .................. : sevmgr
-- PACKAGE_PRETTY_NAME ........... : SEvMgr
-- PACKAGE ....................... : sevmgr
-- PACKAGE_NAME .................. : SEVMGR
-- PACKAGE_BRIEF ................. : C++ Simulation-Oriented Discrete Event Management Library
-- PACKAGE_VERSION ............... : 0.5.0
-- GENERIC_LIB_VERSION ........... : 0.5.0
-- GENERIC_LIB_SOVERSION ......... : 0.5
--
-- ----------------------------------
-- ---     Build Configuration    ---
-- ----------------------------------
-- Modules to build .............. : airrac;rmol;sevmgr
-- Libraries to build/install ..... : airraclib;rmollib;sevmgrlib
-- Binaries to build/install ...... : airrac;rmol;sevmgr_parseInventory;sevmgr
-- Modules to test ............... : sevmgr
-- Binaries to test .............. : InventoryTestSuitetst
--
-- * Module ...................... : sevmgr
--   + Layers to build ........... : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib;rmollib
--   + Libraries to build/install . : sevmgrlib
--   + Executables to build/install : sevmgr_parseInventory;sevmgr
--   + Tests to perform .......... : InventoryTestSuitetst
--
-- BUILD_SHARED_LIBS ............. : ON
-- CMAKE_BUILD_TYPE .............. : Debug
--  * CMAKE_C_FLAGS .............. :
--  * CMAKE_CXX_FLAGS ............ : -Wall -Werror
--  * BUILD_FLAGS ................ :
--  * COMPILE_FLAGS .............. :
-- CMAKE_MODULE_PATH ............. : /home/dan/dev/sim/sevmgr/sevmgrgithub/config/
-- CMAKE_INSTALL_PREFIX .......... : /home/dan/dev/deliveries/sevmgr-0.5.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ............. : 1.7.4
--   - DOXYGEN_EXECUTABLE .......... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ....... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ............ : /usr/bin
--
-- ----------------------------------
-- --- Installation Configuration ---
-- ----------------------------------
-- INSTALL_LIB_DIR ............... : /home/dan/dev/deliveries/sevmgr-0.5.0/lib64
-- INSTALL_BIN_DIR ............... : /home/dan/dev/deliveries/sevmgr-0.5.0/bin
-- INSTALL_INCLUDE_DIR ........... : /home/dan/dev/deliveries/sevmgr-0.5.0/include
-- INSTALL_DATA_DIR .............. : /home/dan/dev/deliveries/sevmgr-0.5.0/share
-- INSTALL_SAMPLE_DIR ............ : /home/dan/dev/deliveries/sevmgr-0.5.0/share/sevmgr/samples
-- INSTALL_DOC ................... : ON
--
-- ----------------------------------
-- ---   Packaging Configuration  ---
-- ----------------------------------
-- CPACK_PACKAGE_CONTACT ......... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR .......... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ......... : 0.5.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/dan/dev/sim/sevmgr/sevmgrgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/dan/dev/sim/sevmgr/sevmgrgithub/COPYING
-- CPACK_GENERATOR ............... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ........ : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : sevmgr-0.5.0
--
-- ---------------------------------
-- ---     External libraries    ---
-- ---------------------------------
--
-- * Boost:
--   - Boost_VERSION ............. : 104600
--   - Boost_LIB_VERSION .......... : 1_46
--   - Boost_HUMAN_VERSION ........ : 1.46.0
--   - Boost_INCLUDE_DIRS ......... : /usr/include
--   - Boost required components .. : regex;program_options;date_time;iostreams;serialization;filesystem;u
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_
--
-- * Readline:
--   - READLINE_VERSION .......... : 6.2
--   - READLINE_INCLUDE_DIR ....... : /usr/include
--   - READLINE_LIBRARY .......... : /usr/lib64/libreadline.so
--
```

```
-- * MySQL:
--   - MYSQL_VERSION ............. : 5.5.14
--   - MYSQL_INCLUDE_DIR .......... : /usr/include/mysql
--   - MYSQL_LIBRARIES ............ : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ............... : 3.0.0
--   - SOCI_INCLUDE_DIR ........... : /usr/include/soci
--   - 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/dan/dev/deliveries/stdair-0.38.0/bin
--   - STDAIR_EXECUTABLES ......... : stdair
--   - STDAIR_LIBRARY_DIRS ........ : /home/dan/dev/deliveries/stdair-0.38.0/lib64
--   - STDAIR_LIBRARIES ........... : stdairlib;stdairuicllib
--   - STDAIR_INCLUDE_DIRS ........ : /home/dan/dev/deliveries/stdair-0.38.0/include
--   - STDAIR_SAMPLE_DIR .......... : /home/dan/dev/deliveries/stdair-0.38.0/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ============================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/dan/dev/sim/sevmgr/sevmgrgithub/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_sevmgr
[  0%] Built target hdr_cfg_airrac
[ 13%] Built target hdr_cfg_rmol
[ 98%] Built target sevmgrlib
[100%] Built target InventoryTestSuitetst
Scanning dependencies of target check_sevmgrtst
Test project /home/dan/dev/sim/sevmgr/sevmgrgithub/build/test/sevmgr
    Start 1: InventoryTestSuitetst
1/1 Test #1: InventoryTestSuitetst ...........   Passed    0.08 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.35 sec
[100%] Built target check_sevmgrtst
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/sevmgrgit
rm -rf build && mkdir build
cd build
```

`to remove everything.`

10 Linking with SEvMgr

10.1 Table of Contents

10.2 Introduction

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

10.3 Dependencies

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

10.3.1 StdAir

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

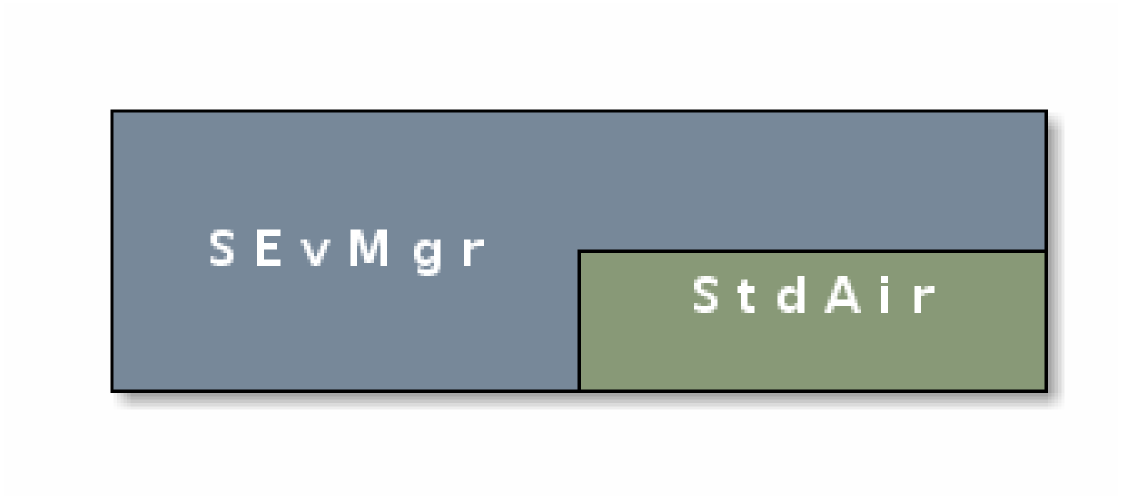


Figure 1: SEvMgr Dependencies

10.4 Using the pkg-config command

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

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

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

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

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

10.5 Using the sevmgr-config script

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

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

```
sevmgr-config --help
```

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

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

10.6 M4 macro for the GNU Autotools

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

- `'SEvMgr_VERSION'` (e.g., defined to 0.23.0)
- `'SEvMgr_CFLAGS'` (e.g., defined to `'-I${prefix}/include'`)
- `'SEvMgr_LIBS'` (e.g., defined to `'-L${prefix}/lib -lsevmgr'`)

10.7 Using SEvMgr with dynamic linking

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

11 Test Rules

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

11.1 The Test File

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

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

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

11.2 The Reference File

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

11.3 Testing IT++ Library

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

```
% make check
```

after successful compilation of the IT++ library.

12 Users Guide

12.1 Table of Contents

12.2 Introduction

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

12.3 Get Started

12.3.1 Get the SEvMgr library

Clone locally the full Git project:

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

12.3.2 Build the SEvMgr project

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

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

12.3.3 Build and Run the Tests

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

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

As a result, you should obtain a similar report:

```
[  0%] Built target hdr_cfg_sevmgr
[ 96%] Built target sevmgrlib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_sevmgrtst
Test project /home/dan/dev/sim/sevmgr/sevmgrgithub/build/test/sevmgr
    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_sevmgrtst
Scanning dependencies of target check
[100%] Built target check
```

12.3.4 Install the SEvMgr Project (Binaries, Documentation)

After the step Build the SEvMgr project, to install the library and its header files, type:

```
cd ~/dev/sim/sevmgrgit
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/sevmgr-0.5.0
```

To generate the SEvMgr project documentation, the commands are:

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

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

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

12.4 Input file of SEvMgr Project

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

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

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

12.5 The schedule BOM Tree

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

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

12.5.1 Build of the schedule BOM tree

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

```
   * Nothing is being done at that stage. The buildSampleBom() method may
```

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

12.5.2 Display of the schedule BOM tree

Note:

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

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

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

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

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

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

```
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-23
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-23, SIN-BKK, 2010-Jan-23, 08:20:00, 2010-Jan-23, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 6.64029e-31
      9, 0, 300, 9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-24
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-24, SIN-BKK, 2010-Jan-24, 08:20:00, 2010-Jan-24, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-25, SIN-BKK, 2010-Jan-25, 08:20:00, 2010-Jan-25, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-26, SIN-BKK, 2010-Jan-26, 08:20:00, 2010-Jan-26, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-27
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-27, SIN-BKK, 2010-Jan-27, 08:20:00, 2010-Jan-27, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
```

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

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

```
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Jan-31
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Jan-31, SIN-BKK, 2010-Jan-31, 08:20:00, 2010-Jan-31, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-01
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-01, SIN-BKK, 2010-Feb-01, 08:20:00, 2010-Feb-01, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-02
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-02, SIN-BKK, 2010-Feb-02, 08:20:00, 2010-Feb-02, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-03
```

```
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-03, SIN-BKK, 2010-Feb-03, 08:20:00, 2010-Feb-03, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-04
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-04, SIN-BKK, 2010-Feb-04, 08:20:00, 2010-Feb-04, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-05
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-05, SIN-BKK, 2010-Feb-05, 08:20:00, 2010-Feb-05, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-06
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-06, SIN-BKK, 2010-Feb-06, 08:20:00, 2010-Feb-06, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
```

```
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-07
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-07, SIN-BKK, 2010-Feb-07, 08:20:00, 2010-Feb-07, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-08
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-08, SIN-BKK, 2010-Feb-08, 08:20:00, 2010-Feb-08, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-09
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-09, SIN-BKK, 2010-Feb-09, 08:20:00, 2010-Feb-09, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
```

```
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-10
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-10, SIN-BKK, 2010-Feb-10, 08:20:00, 2010-Feb-10, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
       9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-11
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-11, SIN-BKK, 2010-Feb-11, 08:20:00, 2010-Feb-11, 11:00:00, 07:40:00
```

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

```
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-13
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-13, SIN-BKK, 2010-Feb-13, 08:20:00, 2010-Feb-13, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-14
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-14, SIN-BKK, 2010-Feb-14, 08:20:00, 2010-Feb-14, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
```

```
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-15
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, 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, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-16
******************************************
******************************************
Leg-Dates:
```

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

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

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

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

```
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-25, SIN-BKK, 2010-Feb-25, 08:20:00, 2010-Feb-25, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-26, SIN-BKK, 2010-Feb-26, 08:20:00, 2010-Feb-26, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-27
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-27, SIN-BKK, 2010-Feb-27, 08:20:00, 2010-Feb-27, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ11, 2010-Feb-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ11 2010-Feb-28, SIN-BKK, 2010-Feb-28, 08:20:00, 2010-Feb-28, 11:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, 0, 0, 0, 0, 300, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-15
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-15, SIN-HND, 2010-Jan-15, 09:20:00, 2010-Jan-15, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 200, 200, 2.082e+121, 5.53287e-48, 5.20
      268e-90, 0, 1.31346e-47, 1.05119e-153, 2.78986e+179, 0, 200, 9, 3.66962e-62, 1.08
      54e-71, 6.74783e-67, 6.9835e-77, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, Y13856, 200 (0), 0, 0, 0, 0, 0 (0),
      0, 0, 0, 0, 0, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-16
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-16, SIN-HND, 2010-Jan-16, 09:20:00, 2010-Jan-16, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 2.63638e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
```

```
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-17
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-17, SIN-HND, 2010-Jan-17, 09:20:00, 2010-Jan-17, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 2.39291e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-18
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-18, SIN-HND, 2010-Jan-18, 09:20:00, 2010-Jan-18, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 2.14469e-319, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-19
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-19, SIN-HND, 2010-Jan-19, 09:20:00, 2010-Jan-19, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-20
******************************************
```

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

```
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-24
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-24, SIN-HND, 2010-Jan-24, 09:20:00, 2010-Jan-24, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-25, SIN-HND, 2010-Jan-25, 09:20:00, 2010-Jan-25, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-26, SIN-HND, 2010-Jan-26, 09:20:00, 2010-Jan-26, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
```

```
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-27
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-27, SIN-HND, 2010-Jan-27, 09:20:00, 2010-Jan-27, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-28, SIN-HND, 2010-Jan-28, 09:20:00, 2010-Jan-28, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-29
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-29, SIN-HND, 2010-Jan-29, 09:20:00, 2010-Jan-29, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-30
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-30, SIN-HND, 2010-Jan-30, 09:20:00, 2010-Jan-30, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Jan-31
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Jan-31, SIN-HND, 2010-Jan-31, 09:20:00, 2010-Jan-31, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
```

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

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

```
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-04
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-04, SIN-HND, 2010-Feb-04, 09:20:00, 2010-Feb-04, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-05
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-05, SIN-HND, 2010-Feb-05, 09:20:00, 2010-Feb-05, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-06
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-06, SIN-HND, 2010-Feb-06, 09:20:00, 2010-Feb-06, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-07
```

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

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

```
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-14
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
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, Comm
      Space, 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, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-15, SIN-HND, 2010-Feb-15, 09:20:00, 2010-Feb-15, 12:00:00, 07:40:00
```

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

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

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

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

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

```
FlightDate: SQ12, 2010-Feb-25
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-25, SIN-HND, 2010-Feb-25, 09:20:00, 2010-Feb-25, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-26
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-26, SIN-HND, 2010-Feb-26, 09:20:00, 2010-Feb-26, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
```

```
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-27
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-27, SIN-HND, 2010-Feb-27, 09:20:00, 2010-Feb-27, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
******************************************
FlightDate: SQ12, 2010-Feb-28
******************************************
******************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
SQ12 2010-Feb-28, SIN-HND, 2010-Feb-28, 09:20:00, 2010-Feb-28, 12:00:00, 07:40:00
      , 0, -05:00:00, 6300, 0,
******************************************
******************************************
```

```
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200,
      9, 0, 0, 0, 0, 0,
******************************************
******************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
******************************************
******************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, 0, 0, 0, 0, 200, 0,
******************************************
******************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0, 0,
       0, 0, 0, 0,
******************************************
```

12.6 Exploring the Predefined BOM Tree

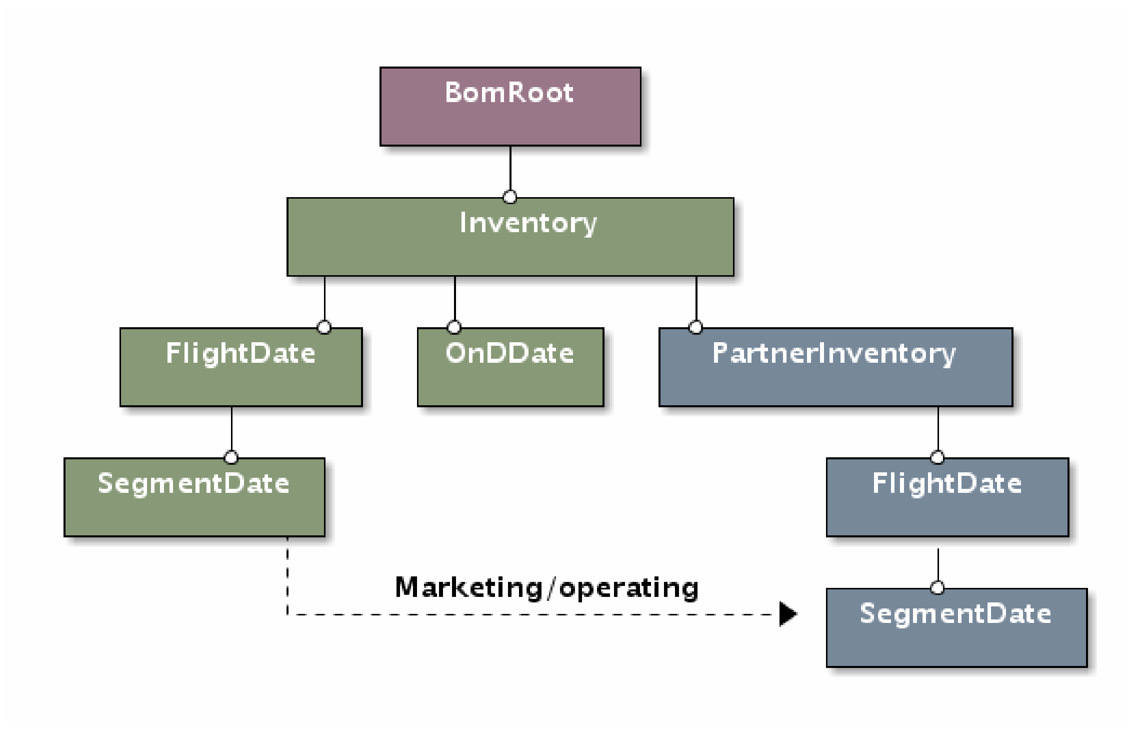


Figure 2: SEvMgr BOM tree

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

12.6.1 Airline Network BOM Tree

- `SEVMGR::ReachableUniverse`
- `SEVMGR::OriginDestinationSet`
- `SEVMGR::SegmentPathPeriod`

12.6.2 Airline Schedule BOM Tree

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

12.7 Extending the BOM Tree

12.8 The travel solution calculation procedure

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

13 Supported Systems

13.1 Table of Contents

13.2 Introduction

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

14 SEvMgr Supported Systems (Previous Releases)

14.1 SEvMgr 3.9.1

14.2 SEvMgr 3.9.0

14.3 SEvMgr 3.8.1

15 Tutorials

15.1 Table of Contents

15.2 Preparing the AirSched Project for Development

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

15.3 Your first networkBuilde

15.3.1 Summary of the different steps

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

First, we 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):

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

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

15.3.2 Result of the Batch Program

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

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

and after the network building:

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

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

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

15.4 Network building with an input file

15.4.1 How to build a network input file?

The objective here is to build a network input file to network 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 :

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

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

The input file is now:

Let us say we have just the Economy cabin "Y" and 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:

15.4.2 Building the BOM tree with an input file

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

15.4.3 Result of the Batch Program

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

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

the last lines of the log output should look like:

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

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

16 Command-Line Test to Demonstrate How To Use Sevmgr elements

```
 */
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
```

```
#include <fstream>
#include <map>
#include <cmath>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE EventQueueManagementTest
#include <boost/test/unit_test.hpp>
#include <boost/shared_ptr.hpp>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/service/Logger.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/config/sevmgr-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("EventQueueManagementTestSuite_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_tes
      ts);
  }

  ~UnitTestConfig() {
  }
};

// Specific type definitions
typedef std::pair<stdair::Count_T, stdair::Count_T> NbOfEventsPair_T;
typedef std::map<const stdair::DemandStreamKeyStr_T,
                 NbOfEventsPair_T> NbOfEventsByDemandStreamMap_T;


// /////////////// 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 (sevmgr_simple_simulation_test) {

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

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

```
// Initialise the Sevmgr service object
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
SEVMGR::SEVMGR_Service sevmgrService (lLogParams);

const bool isQueueDone = sevmgrService.isQueueDone();
BOOST_REQUIRE_MESSAGE (isQueueDone == true,
                       "The event queue should be empty at this step. No "
                       << "insertion done.");

sevmgrService.buildSampleQueue ();

stdair::Count_T lNbOfEvents (sevmgrService.getQueueSize());

BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == false,
                       "The event queue should not be empty at this step. "
                       << "Two insertions done.");

stdair::Count_T idx = 1;
while (sevmgrService.isQueueDone() == false) {

  // Pop the next event out of the event queue
  stdair::EventStruct lEventStruct;
  const stdair::ProgressStatusSet lPPS =
    sevmgrService.popEvent (lEventStruct);

  // DEBUG
  STDAIR_LOG_DEBUG ("Poped event "<< idx << ": '"
                    << lEventStruct.describe() << "'.");
  STDAIR_LOG_DEBUG ("Progresss status: " << lPPS.describe());
  STDAIR_LOG_DEBUG ("Poped event: '"
                    << lEventStruct.describe() << "'.");

  // Iterate
  ++idx;
}

// Compensate for the last iteration
--idx;
// Compared the actual number of popped events with the expected one.
BOOST_REQUIRE_MESSAGE (idx == lNbOfEvents,
                       "Actual number of requests in the queue: "
                       << idx << ". Expected value: " << lNbOfEvents);

BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
                       "The event queue should be empty at this step: "
                       "the two events have been popped.");

STDAIR_LOG_DEBUG ("Re-added the events into the queue");

// Add again the four events into the queue thanks to
// sevmgrService.buildSampleQueue().
// Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
// Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
sevmgrService.buildSampleQueue ();

// Pop the next event out of the event queue
stdair::EventStruct lFirstEventStruct;
const stdair::ProgressStatusSet lFirstPS =
  sevmgrService.popEvent (lFirstEventStruct);

// Extract the corresponding date
const stdair::DateTime_T& lFirstEventDateTime =
  lFirstEventStruct.getEventTime ();
const stdair::Date_T& lFirstRequestDate =
  lFirstEventDateTime.date();
  const stdair::Date_T lExpectedDate (2010, boost::gregorian::Jan, 21);
  BOOST_REQUIRE_MESSAGE (lFirstRequestDate == lExpectedDate,
                         "Date of the first event popped from the queue: "
                         << lFirstRequestDate << ". Should be: "
                         << lExpectedDate << " which is earlier in time.");

  STDAIR_LOG_DEBUG ("Reset the queue");
  sevmgrService.reset();

  BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
                         "The event queue has been reset: it should be empty "
                         << "at this step.");

  STDAIR_LOG_DEBUG ("Re-added the events into the queue one more time");

  // Add again the four events into the queue thanks to
  // sevmgrService.buildSampleQueue().
  // Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
  // Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
  sevmgrService.buildSampleQueue ();

  stdair::EventStruct lBreakPointStruct;
  sevmgrService.run(lBreakPointStruct);
  stdair::EventType::EN_EventType lBreakPointType =
    lBreakPointStruct.getEventType();

  BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
                         "The last event poppped from the queue should be a "
                         << "break point.");

  sevmgrService.run(lBreakPointStruct);
  lBreakPointType = lBreakPointStruct.getEventType();

  BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
                         "The last event poppped from the queue should be a "
                         << "break point.");

  // Extract the corresponding date
  const stdair::DateTime_T& lBPDateTime =
    lBreakPointStruct.getEventTime ();
  const stdair::Date_T& lBPDate =
    lBPDateTime.date();

  const stdair::Date_T lExpectedBPDate (2011, boost::gregorian::May, 14);
  BOOST_REQUIRE_MESSAGE (lBPDate == lExpectedBPDate,
                         "Date of the second break point popped from the queue: "

                         << lBPDate << ". Should be: "
                         << lExpectedBPDate << ".");

  // DEBUG
  STDAIR_LOG_DEBUG ("End of the simulation");

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

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

17 Directory Hierarchy

17.1 Directories

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

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:

std::basic_fstream< char >
std::basic_fstream< wchar_t >
std::basic_ifstream< char >

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 Directory Documentation

22.1 sevmgr/basic/ Directory Reference

Files

- file BasConst.cpp
- file BasConst_EventQueueManager.hpp
- file BasConst_SEVMGR_Service.hpp
- file BasParserTypes.hpp

22.2 sevmgr/batches/ Directory Reference

Files

- file sevmgr_demo.cpp

22.3 sevmgr/bom/ Directory Reference

Files

- file BomJSONExport.cpp
- file BomJSONExport.hpp
- file EventQueue.cpp
- file EventQueue.hpp
- file EventQueueKey.cpp
- file EventQueueKey.hpp
- file EventQueueTypes.hpp

22.4 sevmgr/ui/cmdline/ Directory Reference

Files

- file sevmgr.cpp

22.5 sevmgr/command/ Directory Reference

Files

- file EventQueueManager.cpp
- file EventQueueManager.hpp

22.6 sevmgr/config/ Directory Reference

Files

- file sevmgr-paths.hpp.in

22.7 sevmgr/factory/ Directory Reference

Files

- file FacSEVMGRServiceContext.cpp
- file FacSEVMGRServiceContext.hpp

22.8 sevmgr/python/ Directory Reference

Files

- file pysevmgr.cpp

22.9 sevmgr/service/ Directory Reference

Files

- file SEVMGR_Service.cpp
- file SEVMGR_ServiceContext.cpp
- file SEVMGR_ServiceContext.hpp

22.10 test/sevmgr/ Directory Reference

Files

- file EventQueueManagementTestSuite.cpp

22.11 sevmgr/ Directory Reference

Directories

- directory basic
- directory batches
- directory bom
- directory command
- directory config
- directory factory
- directory python
- directory service
- directory ui

Files

- file SEVMGR_Exceptions.hpp
- file SEVMGR_Service.hpp
- file SEVMGR_Types.hpp

22.12 test/ Directory Reference

Directories

- directory sevmgr

22.13 sevmgr/ui/ Directory Reference

Directories

- directory cmdline

23 Namespace Documentation

23.1 bpt Namespace Reference

Typedefs

- typedef char ptree

23.1.1 Typedef Documentation

23.1.1.1 typedef char bpt::ptree

Definition at line 24 of file BomJSONExport.cpp.

23.2 SEVMGR Namespace Reference

Classes

- class BomJSONExport

  Utility class to export StdAir objects in a JSON format.

- class EventQueue

  Class holding event structures.

- struct EventQueueKey
- class EventQueueManager

  Utility class for Demand and DemandStream objects.

- class FacSEVMGRServiceContext
- struct PYEventQueueManager
- class SEVMGR_ServiceContext

  Class holding the context of the Sevmgr services.

- class SEvMgrException
- class EventQueueException
- class SEVMGR_Service

  class holding the services related to Travel Demand Generation.

Typedefs

- typedef char char_t
- typedef boost::spirit::classic::file_iterator< char_t > iterator_t
- typedef boost::spirit::classic::scanner< iterator_t > scanner_t
- typedef boost::spirit::classic::rule< scanner_t > rule_t
- typedef boost::spirit::classic::int_parser< unsigned int, 10, 1, 1 > int1_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 2, 2 > uint2_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 2 > uint1_2_p_t

- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 3 > uint1_3_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 4, 4 > uint4_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 4 > uint1_4_p_t
- typedef boost::spirit::classic::chset< char_t > chset_t
- typedef boost::spirit::classic::impl::loop_traits< chset_t, unsigned int, unsigned int >::type repeat_-p_t
- typedef boost::spirit::classic::bounded< uint2_p_t, unsigned int > bounded2_p_t
- typedef boost::spirit::classic::bounded< uint1_2_p_t, unsigned int > bounded1_2_p_t
- typedef boost::spirit::classic::bounded< uint1_3_p_t, unsigned int > bounded1_3_p_t
- typedef boost::spirit::classic::bounded< uint4_p_t, unsigned int > bounded4_p_t
- typedef boost::spirit::classic::bounded< uint1_4_p_t, unsigned int > bounded1_4_p_t
- typedef std::list< EventQueue * > EventQueueList_T
- typedef std::map< const stdair::MapKey_T, EventQueue * > EventQueueMap_T
- typedef boost::shared_ptr< SEVMGR_Service > SEVMGR_ServicePtr_T
- typedef std::string EventQueueID_T
- typedef std::map< stdair::EventType::EN_EventType, stdair::ProgressStatus > ProgressStatusMap_T

Functions

- const EventQueueID_T DEFAULT_EVENT_QUEUE_ID ("EQ01")

Variables

- const EventQueueID_T DEFAULT_EVENT_QUEUE_ID

23.2.1 Typedef Documentation

23.2.1.1 typedef char SEVMGR::char_t

Definition at line 31 of file BasParserTypes.hpp.

23.2.1.2 typedef boost::spirit::classic::file_iterator<char_t> SEVMGR::iterator_t

Definition at line 35 of file BasParserTypes.hpp.

23.2.1.3 typedef boost::spirit::classic::scanner<iterator_t> SEVMGR::scanner_t

Definition at line 36 of file BasParserTypes.hpp.

23.2.1.4 typedef boost::spirit::classic::rule<scanner_t> SEVMGR::rule_t

Definition at line 37 of file BasParserTypes.hpp.

23.2.1.5 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> SEVMGR::int1_p_t

1-digit-integer parser

Definition at line 45 of file BasParserTypes.hpp.

23.2.1.6 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> SEVMGR::uint2_p_t

2-digit-integer parser

Definition at line 48 of file BasParserTypes.hpp.

23.2.1.7 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> SEVMGR::uint1_2_p_t

Up-to-2-digit-integer parser

Definition at line 51 of file BasParserTypes.hpp.

23.2.1.8 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3> SEVMGR::uint1_3_p_t

Up-to-3-digit-integer parser

Definition at line 54 of file BasParserTypes.hpp.

23.2.1.9 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> SEVMGR::uint4_p_t

4-digit-integer parser

Definition at line 57 of file BasParserTypes.hpp.

23.2.1.10 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4> SEVMGR::uint1_4_p_t

Up-to-4-digit-integer parser

Definition at line 60 of file BasParserTypes.hpp.

23.2.1.11 typedef boost::spirit::classic::chset<char_t> SEVMGR::chset_t

character set

Definition at line 63 of file BasParserTypes.hpp.

23.2.1.12 typedef boost::spirit::classic::impl::loop_traits<chset_t, unsigned int, unsigned int>::type SEVMGR::repeat_p_t

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

Definition at line 69 of file BasParserTypes.hpp.

23.2.1.13 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> SEVMGR::bounded2_p_t

Bounded-number-of-integers parser

Definition at line 72 of file BasParserTypes.hpp.

23.2.1.14 typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int> SEVMGR::bounded1_2_p_t

Definition at line 73 of file BasParserTypes.hpp.

23.2.1.15 typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int> SEVMGR::bounded1_3_p_t

Definition at line 74 of file BasParserTypes.hpp.

23.2.1.16 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> SEVMGR::bounded4_p_t

Definition at line 75 of file BasParserTypes.hpp.

23.2.1.17 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int> SEVMGR::bounded1_4_p_t

Definition at line 76 of file BasParserTypes.hpp.

23.2.1.18 typedef std::list<EventQueue∗> SEVMGR::EventQueueList_T

Define the EventQueue list.

Definition at line 17 of file EventQueueTypes.hpp.

23.2.1.19 typedef std::map<const stdair::MapKey_T, EventQueue∗> SEVMGR::EventQueueMap_T

Define the EventQueue map.

Definition at line 23 of file EventQueueTypes.hpp.

23.2.1.20 typedef boost::shared_ptr<SEVMGR_Service> SEVMGR::SEVMGR_ServicePtr_T

(Smart) Pointer on the SEvMgr service handler.

Definition at line 18 of file SEVMGR_Types.hpp.

23.2.1.21 typedef std::string SEVMGR::EventQueueID_T

Define an ID for an EventQueue object.

Definition at line 27 of file SEVMGR_Types.hpp.

23.2.1.22 typedef std::map<stdair::EventType::EN_EventType, stdair::ProgressStatus> SEVMGR::ProgressStatusMap_T

Definition of the (STL) map of ProgressStatus structures, one for each event type (e.g., booking request, optimisation notification).

Definition at line 35 of file SEVMGR_Types.hpp.

23.2.2 Function Documentation

23.2.2.1 const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID ("EQ01")

Default name for the SEVMGR_Service. Default ID for the event queue.

23.2.3 Variable Documentation

23.2.3.1 const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID

Default ID for the event queue.

23.3 stdair Namespace Reference

Forward declarations.

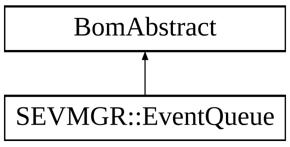
23.3.1 Detailed Description

Forward declarations.

24 Class Documentation

24.1 BomAbstract Class Reference

Inheritance diagram for BomAbstract::



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

- sevmgr/bom/EventQueue.hpp

24.2 SEVMGR::BomJSONExport Class Reference

Utility class to export StdAir objects in a JSON format.

`#include <sevmgr/bom/BomJSONExport.hpp>`

Static Public Member Functions

- static void jsonExportEventQueue (stdair::STDAIR_ServicePtr_T &, std::ostream &, const EventQueue &, const stdair::EventType::EN_EventType &)

24.2.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

Definition at line 34 of file BomJSONExport.hpp.

24.2.2 Member Function Documentation

24.2.2.1 void SEVMGR::BomJSONExport::jsonExportEventQueue (stdair::STDAIR_ServicePtr_T & *ioSTDAIR_ServicePtr*, std::ostream & *oStream*, const EventQueue & *iEventQueue*, const stdair::EventType::EN_EventType & *iEventType*) `[static]`

Export (dump in the underlying output log stream and in JSON format) the event struct objects contained in the event queue.

STDAIR_ServicePtr_T& Pointer on the StdAir service handler.

Parameters:

std::ostream& Output stream in which the events should be logged/dumped.

const EventQueue& Events queue to be stored in JSON-ified format.

const stdair::EventType::EN_EventType& Filter to select objects with a certain event type.

Definition at line 32 of file BomJSONExport.cpp.

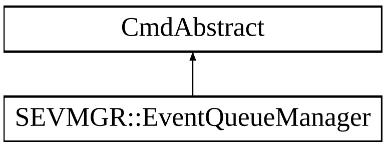
References SEVMGR::EventQueue::getEventList().

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

- sevmgr/bom/BomJSONExport.hpp
- sevmgr/bom/BomJSONExport.cpp

24.3 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract::



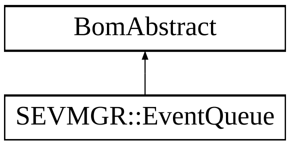
The documentation for this class was generated from the following file:

- sevmgr/command/EventQueueManager.hpp

24.4 SEVMGR::EventQueue Class Reference

Class holding event structures.

`#include <sevmgr/bom/EventQueue.hpp>`Inheritance diagram for SEVMGR::EventQueue::



Public Types

- typedef EventQueueKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract ∗const getParent () const
- const stdair::EventList_T & getEventList () const
- const stdair::HolderMap_T & getHolderMap () const
- const stdair::ProgressStatus & getStatus () const
- const stdair::Count_T & getCurrentNbOfEvents () const
- const stdair::Count_T & getExpectedTotalNbOfEvents () const
- const stdair::Count_T & getActualTotalNbOfEvents () const
- const stdair::ProgressStatus & getStatus (const stdair::EventType::EN_EventType &) const
- const stdair::Count_T & getCurrentNbOfEvents (const stdair::EventType::EN_EventType &) const
- const stdair::Count_T & getExpectedTotalNbOfEvents (const stdair::EventType::EN_EventType &) const
- const stdair::Count_T & getActualTotalNbOfEvents (const stdair::EventType::EN_EventType &) const
- bool hasProgressStatus (const stdair::EventType::EN_EventType &) const
- void setStatus (const stdair::ProgressStatus &iProgressStatus)
- void setStatus (const stdair::Count_T &iCurrentNbOfEvents, const stdair::Count_T &iExpectedTotalNbOfEvents, const stdair::Count_T &iActualTotalNbOfEvents)
- void setStatus (const stdair::Count_T &iCurrentNbOfEvents, const stdair::Count_T &iActualTotalNbOfEvents)
- void setCurrentNbOfEvents (const stdair::Count_T &iCurrentNbOfEvents)

- void setExpectedTotalNbOfEvents (const stdair::Count_T &iExpectedTotalNbOfEvents)
- void setStatus (const stdair::EventType::EN_EventType &iType, const stdair::ProgressStatus &iProgressStatus)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- std::string list () const
- std::string list (const stdair::EventType::EN_EventType &) const
- const std::string describeKey () const
- std::string display () const
- void reset ()
- stdair::ProgressStatusSet popEvent (stdair::EventStruct &)
- bool addEvent (stdair::EventStruct &)
- bool hasEventDateTime (const stdair::DateTime_T &)
- bool isQueueDone () const
- void addStatus (const stdair::EventType::EN_EventType &, const stdair::NbOfRequests_T &iExpectedTotalNbOfEvents)
- void updateStatus (const stdair::EventType::EN_EventType &, const stdair::ProgressStatus &iProgressStatus)
- void updateStatus (const stdair::EventType::EN_EventType &, const stdair::NbOfEvents_T &iActualTotalNbOfEvents)
- stdair::ProgressPercentage_T calculateProgress () const
- stdair::ProgressPercentage_T calculateProgress (const stdair::EventType::EN_EventType &) const
- stdair::Count_T getQueueSize () const
- bool isQueueEmpty () const

Protected Member Functions

- EventQueue (const Key_T &)
- EventQueue (const EventQueue &)
- ∼EventQueue ()

Protected Attributes

- Key_T _key
- BomAbstract ∗ _parent
- stdair::HolderMap_T _holderMap
- stdair::EventList_T _eventList
- stdair::ProgressStatus _progressStatus
- ProgressStatusMap_T _progressStatusMap

Friends

- class stdair::FacBom
- class stdair::FacBomManager

24.4.1 Detailed Description

Class holding event structures. Event types may be:

- booking requests,
- optimisation notifications,
- (simulation) break point,
- schedule changes.

The event content would be, respectively:

- a demand stream (generating booking requests),
- a DCP rule (generation optimisation notifications),
- a break point rule (generating simulation break points),
- a schedule update (generating schedule changes).

The EventQueue object keeps track of the simulation progress, overall and broken down (independently) both by event type and by content key. Following is a full example:

- Break down by event type:
  - Booking request: 9 events out of {expected: 20, actual: 20}
  - Optimisation notification: 7 events out of {expected: 32, actual: 32}
- Break down by content key:
  - "SIN-BKK" demand stream: 5 events out of {expected: 10, actual: 11}
  - "SIN-NRT" demand stream: 4 events out of {expected: 10, actual: 9}
  - "SQ 12" DCP rule: 2 events out of {expected: 16, actual: 16}
  - "SQ 25" DCP rule: 5 events out of {expected: 16, actual: 16}
- Overall status: 16 events out of {expected: 52, actual: 52}

Definition at line 68 of file EventQueue.hpp.

24.4.2 Member Typedef Documentation

24.4.2.1 typedef EventQueueKey SEVMGR::EventQueue::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 77 of file EventQueue.hpp.

24.4.3 Constructor & Destructor Documentation

24.4.3.1 SEVMGR::EventQueue::EventQueue (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 25 of file EventQueue.cpp.

24.4.3.2 SEVMGR::EventQueue::EventQueue (const EventQueue & *iEventQueue*) `[protected]`

Default copy constructor.

Definition at line 32 of file EventQueue.cpp.

24.4.3.3 SEVMGR::EventQueue::∼EventQueue () `[protected]`

Destructor.

Definition at line 40 of file EventQueue.cpp.

References _eventList.

24.4.4 Member Function Documentation

24.4.4.1 const Key_T& SEVMGR::EventQueue::getKey () const `[inline]`

Get the event queue key.

Definition at line 83 of file EventQueue.hpp.

References _key.

24.4.4.2 BomAbstract∗ const SEVMGR::EventQueue::getParent () const `[inline]`

Get the parent object.

Definition at line 88 of file EventQueue.hpp.

References _parent.

24.4.4.3 const stdair::EventList_T& SEVMGR::EventQueue::getEventList () const `[inline]`

Get the list of events.

Definition at line 93 of file EventQueue.hpp.

References _eventList.

Referenced by SEVMGR::BomJSONExport::jsonExportEventQueue().

24.4.4.4 const stdair::HolderMap_T& SEVMGR::EventQueue::getHolderMap () const `[inline]`

Get the map of children holders.

Definition at line 98 of file EventQueue.hpp.

References _holderMap.

24.4.4.5 const stdair::ProgressStatus& SEVMGR::EventQueue::getStatus () const `[inline]`

Get the overall progress status (for the whole event queue).

Definition at line 103 of file EventQueue.hpp.

References _progressStatus.

Referenced by popEvent().

24.4.4.6 const stdair::Count_T& SEVMGR::EventQueue::getCurrentNbOfEvents () const `[inline]`

Get the current number of events (for the whole event queue).

Definition at line 107 of file EventQueue.hpp.

References _progressStatus.

24.4.4.7 const stdair::Count_T& SEVMGR::EventQueue::getExpectedTotalNbOfEvents () const `[inline]`

Get the expected total number of events (for the whole event queue).

Definition at line 111 of file EventQueue.hpp.

References _progressStatus.

24.4.4.8 const stdair::Count_T& SEVMGR::EventQueue::getActualTotalNbOfEvents () const `[inline]`

Get the actual total number of events (for the whole event queue).

Definition at line 115 of file EventQueue.hpp.

References _progressStatus.

24.4.4.9 const stdair::ProgressStatus & SEVMGR::EventQueue::getStatus (const stdair::EventType::EN_EventType & *iType*) const

Get the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 327 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.10 const stdair::Count_T & SEVMGR::EventQueue::getCurrentNbOfEvents (const stdair::EventType::EN_EventType & *iType*) const

Get the current number of events for the given event type.

Definition at line 157 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.11 const stdair::Count_T & SEVMGR::EventQueue::getExpectedTotalNbOfEvents (const stdair::EventType::EN_EventType & *iType*) const

Get the expected total number of events for the given event type.

Definition at line 176 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.12 const stdair::Count_T & SEVMGR::EventQueue::getActualTotalNbOfEvents (const stdair::EventType::EN_EventType & *iType*) const

Get the actual total number of events for the given event type.

Definition at line 198 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.13 bool SEVMGR::EventQueue::hasProgressStatus (const stdair::EventType::EN_-EventType & *iType*) const

Check if the event queue has already a progress status for the given event type

Definition at line 136 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.14 void SEVMGR::EventQueue::setStatus (const stdair::ProgressStatus & *iProgressStatus*) `[inline]`

Set/update the progress status.

Definition at line 141 of file EventQueue.hpp.

References _progressStatus.

Referenced by popEvent().

24.4.4.15 void SEVMGR::EventQueue::setStatus (const stdair::Count_T & *iCurrentNbOfEvents*, const stdair::Count_T & *iExpectedTotalNbOfEvents*, const stdair::Count_T & *iActualTotalNbOfEvents*) `[inline]`

Set/update the progress status.

Definition at line 145 of file EventQueue.hpp.

References _progressStatus.

24.4.4.16 void SEVMGR::EventQueue::setStatus (const stdair::Count_T & *iCurrentNbOfEvents*, const stdair::Count_T & *iActualTotalNbOfEvents*) `[inline]`

Set/update the progress status.

Definition at line 153 of file EventQueue.hpp.

References _progressStatus.

24.4.4.17 void SEVMGR::EventQueue::setCurrentNbOfEvents (const stdair::Count_T & *iCurrentNbOfEvents*) `[inline]`

Set the current number of events (for the whole event queue).

Definition at line 159 of file EventQueue.hpp.

References _progressStatus.

24.4.4.18 void SEVMGR::EventQueue::setExpectedTotalNbOfEvents (const stdair::Count_T & *iExpectedTotalNbOfEvents*) `[inline]`

Set the expected total number of events (for the whole event queue).

Definition at line 163 of file EventQueue.hpp.

References _progressStatus.

24.4.4.19 void SEVMGR::EventQueue::setStatus (const stdair::EventType::EN_EventType & *iType*, const stdair::ProgressStatus & *iProgressStatus*)

Set the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 311 of file EventQueue.cpp.

References _progressStatusMap.

24.4.4.20 void SEVMGR::EventQueue::toStream (std::ostream & *ioOut*) const `[inline]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 182 of file EventQueue.hpp.

References toString().

24.4.4.21 void SEVMGR::EventQueue::fromStream (std::istream & *ioIn*) `[inline]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 191 of file EventQueue.hpp.

24.4.4.22 std::string SEVMGR::EventQueue::toString () const

Get the serialised version of the Business Object.

Definition at line 45 of file EventQueue.cpp.

References _eventList, and _progressStatus.

Referenced by display(), list(), toStream(), and updateStatus().

24.4.4.23 std::string SEVMGR::EventQueue::list () const

Get the event list decsription.

Definition at line 64 of file EventQueue.cpp.

References _eventList, describeKey(), and toString().

24.4.4.24 std::string SEVMGR::EventQueue::list (const stdair::EventType::EN_EventType & *iType*) const

Get the event list decsription for a given event type

Definition at line 82 of file EventQueue.cpp.

References _eventList, describeKey(), and toString().

24.4.4.25 const std::string SEVMGR::EventQueue::describeKey () const `[inline]`

Get a string describing the key.

Definition at line 213 of file EventQueue.hpp.

References _key, and SEVMGR::EventQueueKey::toString().

Referenced by list(), and popEvent().

24.4.4.26 std::string SEVMGR::EventQueue::display () const

Definition at line 55 of file EventQueue.cpp.

References toString().

Referenced by calculateProgress(), getActualTotalNbOfEvents(), getCurrentNbOfEvents(), getExpected-TotalNbOfEvents(), getStatus(), and hasProgressStatus().

24.4.4.27 void SEVMGR::EventQueue::reset ()

Reset the event queue.

The event queue is fully emptied.

Definition at line 118 of file EventQueue.cpp.

References _eventList, _progressStatus, and _progressStatusMap.

24.4.4.28 stdair::ProgressStatusSet SEVMGR::EventQueue::popEvent (stdair::EventStruct & *ioEventStruct*)

Pop the next coming (in time) event, and remove it from the event queue.

- The next coming (in time) event corresponds to the event having the earliest date-time stamp. In other words, it is the first/front element of the event queue.
- That (first) event/element is then removed from the event queue

- The progress status is updated for the corresponding event generator.

Definition at line 368 of file EventQueue.cpp.

References _eventList, _progressStatus, describeKey(), getStatus(), and setStatus().

24.4.4.29 bool SEVMGR::EventQueue::addEvent (stdair::EventStruct & *ioEventStruct*)

Add event.

If there already is an event with the same date-time, move the given event one nanosecond forward, and retry the insertion until it succeeds.

That method:

- first adds the event structure in the dedicated list,
- then retrieves the corresponding demand stream,
- and update accordingly the corresponding progress statuses.

Parameters:

stdair::EventStruct& The reference on EventStruct is not constant, because the EventStruct object can be altered: its date-time stamp can be changed accordingly to the location where it has been inserted in the event queue.

Definition at line 436 of file EventQueue.cpp.

References _eventList.

24.4.4.30 bool SEVMGR::EventQueue::hasEventDateTime (const stdair::DateTime_T & *iDateTime*)

Find the event with the given date time, if such event existed.

Definition at line 469 of file EventQueue.cpp.

References _eventList.

24.4.4.31 bool SEVMGR::EventQueue::isQueueDone () const

States whether the event queue has reached the end.

For now, that method states whether the event queue is empty.

Definition at line 112 of file EventQueue.cpp.

References _eventList, and isQueueEmpty().

24.4.4.32 void SEVMGR::EventQueue::addStatus (const stdair::EventType::EN_EventType &, const stdair::NbOfRequests_T & *iExpectedTotalNbOfEvents*)

Initialise the progress statuses for the given event type (e.g., request, snapshot).

The progress status is actually a pair of counters:

- The current number of (already generated) events, for the given event type. That number is initialised to 0 (no event has been generated yet).
- The total number of events (to be generated), also for the given event type.

24.4.4.33 void SEVMGR::EventQueue::updateStatus (const stdair::EventType::EN_EventType & *iType*, const stdair::ProgressStatus & *iProgressStatus*)

Set/update the progress status for the corresponding event type (e.g., booking request, optimisation notification, schedule change, break point).

If there is no ProgressStatus object for that event type yet, one is inserted. Otherwise, the ProgressStatus object is updated.

Definition at line 216 of file EventQueue.cpp.

References _progressStatusMap, and toString().

24.4.4.34 void SEVMGR::EventQueue::updateStatus (const stdair::EventType::EN_EventType & *iType*, const stdair::NbOfEvents_T & *iActualTotalNbOfEvents*)

Update the progress statuses for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

The progress status is actually a pair of counters:

- The current number of (already generated) events, for the given event type. That number is initialised to 0 (no event has been generated yet).
- The total number of events (to be generated), also for the given event type.

Definition at line 282 of file EventQueue.cpp.

References _progressStatus, and _progressStatusMap.

24.4.4.35 stdair::ProgressPercentage_T SEVMGR::EventQueue::calculateProgress () const `[inline]`

Calculate the progress status.

The progress is status is the ratio of:

- the current number of events, summed over all the demand streams,
- over the total number of events, also summed over all the demand streams.

Definition at line 338 of file EventQueue.hpp.

References _progressStatus.

24.4.4.36 stdair::ProgressPercentage_T SEVMGR::EventQueue::calculateProgress (const stdair::EventType::EN_EventType & *iType*) const

Calculate the progress status.

The progress is status is the ratio of:

- the current number of events, summed over all the demand streams,
- over the total number of events, also summed over all the demand streams.

Definition at line 350 of file EventQueue.cpp.

References _progressStatusMap, and display().

24.4.4.37 stdair::Count_T SEVMGR::EventQueue::getQueueSize () const

Queue size

Definition at line 102 of file EventQueue.cpp.

References _eventList.

24.4.4.38 bool SEVMGR::EventQueue::isQueueEmpty () const

Is queue empty

Definition at line 107 of file EventQueue.cpp.

References _eventList.

Referenced by isQueueDone().

24.4.5 Friends And Related Function Documentation

24.4.5.1 friend class stdair::FacBom `[friend]`

Definition at line 69 of file EventQueue.hpp.

24.4.5.2 friend class stdair::FacBomManager `[friend]`

Definition at line 70 of file EventQueue.hpp.

24.4.6 Member Data Documentation

24.4.6.1 Key_T SEVMGR::EventQueue::_key `[protected]`

Primary key (ID).

Definition at line 382 of file EventQueue.hpp.

Referenced by describeKey(), and getKey().

24.4.6.2 BomAbstract* SEVMGR::EventQueue::_parent `[protected]`

Pointer on the parent class (BomRoot).

Definition at line 387 of file EventQueue.hpp.

Referenced by getParent().

24.4.6.3 stdair::HolderMap_T SEVMGR::EventQueue::_holderMap `[protected]`

Map holding the children (e.g., DemandStream objects for booking requests, DCPRule objects for optimisation notifications).

Definition at line 394 of file EventQueue.hpp.

Referenced by getHolderMap().

24.4.6.4 stdair::EventList_T SEVMGR::EventQueue::_eventList `[protected]`

List of events.

Definition at line 399 of file EventQueue.hpp.

Referenced by addEvent(), getEventList(), getQueueSize(), hasEventDateTime(), isQueueDone(), isQueueEmpty(), list(), popEvent(), reset(), toString(), and ∼EventQueue().

24.4.6.5 stdair::ProgressStatus SEVMGR::EventQueue::_progressStatus `[protected]`

Counters holding the overall progress status.

Definition at line 404 of file EventQueue.hpp.

Referenced by calculateProgress(), getActualTotalNbOfEvents(), getCurrentNbOfEvents(), getExpectedTotalNbOfEvents(), getStatus(), popEvent(), reset(), setCurrentNbOfEvents(), setExpectedTotalNbOfEvents(), setStatus(), toString(), and updateStatus().

24.4.6.6 ProgressStatusMap_T SEVMGR::EventQueue::_progressStatusMap `[protected]`

Counters holding the overall progress status, for each event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 411 of file EventQueue.hpp.

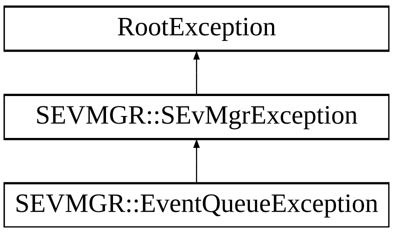
Referenced by calculateProgress(), getActualTotalNbOfEvents(), getCurrentNbOfEvents(), getExpectedTotalNbOfEvents(), getStatus(), hasProgressStatus(), reset(), setStatus(), and updateStatus().

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

- sevmgr/bom/EventQueue.hpp
- sevmgr/bom/EventQueue.cpp

24.5 SEVMGR::EventQueueException Class Reference

`#include <sevmgr/SEVMGR_Exceptions.hpp>`Inheritance diagram for SEVMGR::EventQueueException::



Public Member Functions

- EventQueueException (const std::string &iWhat)

24.5.1 Detailed Description

EventQueue.

Definition at line 28 of file SEVMGR_Exceptions.hpp.

24.5.2 Constructor & Destructor Documentation

24.5.2.1 SEVMGR::EventQueueException::EventQueueException (const std::string & *iWhat*) `[inline]`

Constructor.

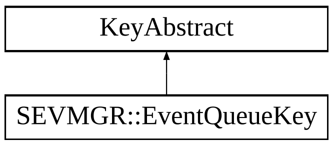
Definition at line 31 of file SEVMGR_Exceptions.hpp.

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

- sevmgr/SEVMGR_Exceptions.hpp

24.6 SEVMGR::EventQueueKey Struct Reference

`#include <sevmgr/bom/EventQueueKey.hpp>`Inheritance diagram for SEVMGR::EventQueueKey::



Public Member Functions

- EventQueueKey (const EventQueueID_T &)
- EventQueueKey (const EventQueueKey &)
- ∼EventQueueKey ()
- const EventQueueID_T & getEventQueueID () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

24.6.1 Detailed Description

Key of eventqueue.

Definition at line 17 of file EventQueueKey.hpp.

24.6.2 Constructor & Destructor Documentation

24.6.2.1 SEVMGR::EventQueueKey::EventQueueKey (const EventQueueID_T & *iEventQueueID*)

Constructors.

Definition at line 12 of file EventQueueKey.cpp.

24.6.2.2 SEVMGR::EventQueueKey::EventQueueKey (const EventQueueKey & *iKey*)

Definition at line 16 of file EventQueueKey.cpp.

24.6.2.3 SEVMGR::EventQueueKey::∼EventQueueKey ()

Destructor.

Definition at line 21 of file EventQueueKey.cpp.

24.6.3 Member Function Documentation

24.6.3.1 const EventQueueID_T& SEVMGR::EventQueueKey::getEventQueueID () const `[inline]`

Get the ID of the EventQueue object.

Definition at line 33 of file EventQueueKey.hpp.

24.6.3.2 void SEVMGR::EventQueueKey::toStream (std::ostream & *ioOut*) const

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Definition at line 25 of file EventQueueKey.cpp.

References toString().

24.6.3.3 void SEVMGR::EventQueueKey::fromStream (std::istream & *ioIn*)

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Definition at line 30 of file EventQueueKey.cpp.

24.6.3.4 const std::string SEVMGR::EventQueueKey::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 34 of file EventQueueKey.cpp.

Referenced by SEVMGR::EventQueue::describeKey(), and toStream().

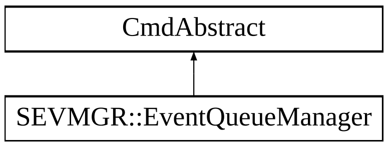
The documentation for this struct was generated from the following files:

- sevmgr/bom/EventQueueKey.hpp
- sevmgr/bom/EventQueueKey.cpp

24.7 SEVMGR::EventQueueManager Class Reference

Utility class for Demand and DemandStream objects.

`#include <sevmgr/command/EventQueueManager.hpp>`Inheritance diagram for SEVMGR::EventQueueManager::



Friends

- class SEVMGR_Service

24.7.1 Detailed Description

Utility class for Demand and DemandStream objects.

Definition at line 27 of file EventQueueManager.hpp.

24.7.2 Friends And Related Function Documentation

24.7.2.1 friend class SEVMGR_Service `[friend]`

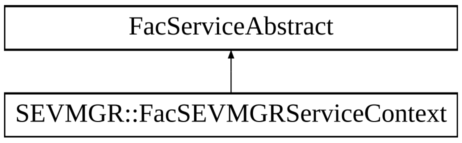
Definition at line 28 of file EventQueueManager.hpp.

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

- sevmgr/command/EventQueueManager.hpp
- sevmgr/command/EventQueueManager.cpp

24.8 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract::

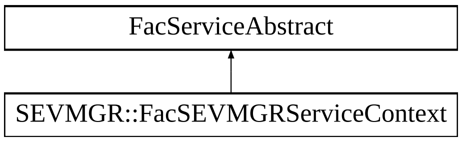


The documentation for this class was generated from the following file:

- sevmgr/factory/FacSEVMGRServiceContext.hpp

24.9 SEVMGR::FacSEVMGRServiceContext Class Reference

`#include <sevmgr/factory/FacSEVMGRServiceContext.hpp>`Inheritance diagram for SEVMGR::FacSEVMGRServiceContext::



Public Member Functions

- ~FacSEVMGRServiceContext ()
- SEVMGR_ServiceContext & create ()

Static Public Member Functions

- static FacSEVMGRServiceContext & instance ()

Protected Member Functions

- FacSEVMGRServiceContext ()

24.9.1 Detailed Description

Factory for Bucket.

Definition at line 18 of file FacSEVMGRServiceContext.hpp.

24.9.2 Constructor & Destructor Documentation

24.9.2.1 SEVMGR::FacSEVMGRServiceContext::~FacSEVMGRServiceContext ()

Destructor.

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

Definition at line 17 of file FacSEVMGRServiceContext.cpp.

24.9.2.2 SEVMGR::FacSEVMGRServiceContext::FacSEVMGRServiceContext () `[inline, protected]`

Default Constructor.

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

Definition at line 42 of file FacSEVMGRServiceContext.hpp.

Referenced by instance().

24.9.3 Member Function Documentation

24.9.3.1 FacSEVMGRServiceContext & SEVMGR::FacSEVMGRServiceContext::instance () `[static]`

Provide the unique instance.

The singleton is instantiated when first used

Returns:

FacSEVMGRServiceContext&

Definition at line 22 of file FacSEVMGRServiceContext.cpp.

References FacSEVMGRServiceContext().

24.9.3.2 SEVMGR_ServiceContext & SEVMGR::FacSEVMGRServiceContext::create ()

Create a new SEVMGR_ServiceContext object.

This new object is added to the list of instantiated objects.

Returns:

SEVMGR_ServiceContext& The newly created object.

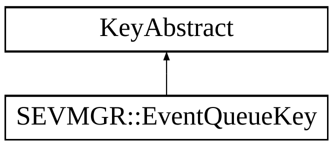
Definition at line 34 of file FacSEVMGRServiceContext.cpp.

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

- sevmgr/factory/FacSEVMGRServiceContext.hpp
- sevmgr/factory/FacSEVMGRServiceContext.cpp

24.10 KeyAbstract Class Reference

Inheritance diagram for KeyAbstract::



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

- sevmgr/bom/EventQueueKey.hpp

24.11 SEVMGR::PYEventQueueManager Struct Reference

Public Member Functions

- std::string sevmgr ()
- PYEventQueueManager ()
- PYEventQueueManager (const PYEventQueueManager &iPYEventQueueManager)
- ∼PYEventQueueManager ()
- bool init (const std::string &iLogFilepath, const std::string &iDBUser, const std::string &iDBPasswd, const std::string &iDBHost, const std::string &iDBPort, const std::string &iDBDBName)

24.11.1 Detailed Description

Definition at line 22 of file pysevmgr.cpp.

24.11.2 Constructor & Destructor Documentation

24.11.2.1 SEVMGR::PYEventQueueManager::PYEventQueueManager () `[inline]`

Default constructor.

Definition at line 76 of file pysevmgr.cpp.

24.11.2.2 SEVMGR::PYEventQueueManager::PYEventQueueManager (const PYEventQueueManager & *iPYEventQueueManager*) `[inline]`

Default copy constructor.

Definition at line 80 of file pysevmgr.cpp.

24.11.2.3 SEVMGR::PYEventQueueManager::∼PYEventQueueManager () `[inline]`

Default constructor.

Definition at line 86 of file pysevmgr.cpp.

24.11.3 Member Function Documentation

24.11.3.1 std::string SEVMGR::PYEventQueueManager::sevmgr () `[inline]`

Wrapper around the travel demand generation use case.

Definition at line 25 of file pysevmgr.cpp.

Referenced by BOOST_PYTHON_MODULE().

24.11.3.2 bool SEVMGR::PYEventQueueManager::init (const std::string & *iLogFilepath*, const std::string & *iDBUser*, const std::string & *iDBPasswd*, const std::string & *iDBHost*, const std::string & *iDBPort*, const std::string & *iDBDBName*) `[inline]`

Wrapper around the search use case.

Definition at line 92 of file pysevmgr.cpp.

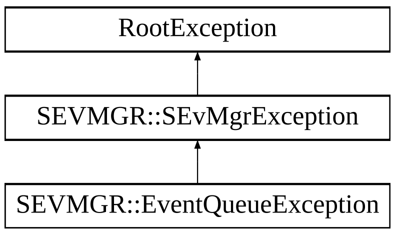
Referenced by BOOST_PYTHON_MODULE().

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

- sevmgr/python/pysevmgr.cpp

24.12 RootException Class Reference

Inheritance diagram for RootException::

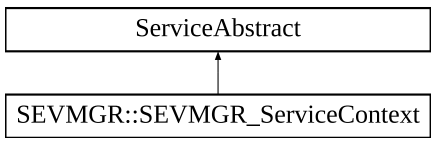


The documentation for this class was generated from the following file:

- sevmgr/SEVMGR_Exceptions.hpp

24.13 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract::



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

- sevmgr/service/SEVMGR_ServiceContext.hpp

24.14 SEVMGR::SEVMGR_Service Class Reference

class holding the services related to Travel Demand Generation.

```
#include <sevmgr/SEVMGR_Service.hpp>
```

Public Member Functions

- SEVMGR_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)

  Constructor.

- SEVMGR_Service (const stdair::BasLogParams &)
- SEVMGR_Service (stdair::STDAIR_ServicePtr_T)
- ~SEVMGR_Service ()
- void buildSampleQueue ()
- stdair::BookingRequestStruct buildSampleBookingRequest (const bool isForCRS=false)
- stdair::ProgressStatusSet popEvent (stdair::EventStruct &) const
- void run (stdair::EventStruct &) const
- bool select (stdair::EventStruct &, const stdair::DateTime_T &) const
- template<class EventGenerator >
  void addEventGenerator (EventGenerator &iEventGenerator) const
- void addEvent (stdair::EventStruct &) const
- void reset () const
- void updateStatus (const stdair::EventType::EN_EventType &, const stdair::Count_T &) const
- void addStatus (const stdair::EventType::EN_EventType &, const stdair::Count_T &) const
- bool isQueueDone () const
- bool hasProgressStatus (const stdair::EventType::EN_EventType &) const
- EventQueue & getEventQueue () const
- const stdair::Count_T & getQueueSize () const
- template<class EventGenerator , class Key >
  EventGenerator & getEventGenerator (const Key &iKey) const
- template<class EventGenerator , class Key >
  bool hasEventGenerator (const Key &iKey) const
- template<class EventGenerator >
  const std::list< EventGenerator * > getEventGeneratorList () const
- template<class EventGenerator >
  bool hasEventGeneratorList () const
- const stdair::Count_T & getExpectedTotalNumberOfEventsToBeGenerated () const
- const stdair::Count_T & getExpectedTotalNumberOfEventsToBeGenerated (const stdair::EventType::EN_EventType &) const
- const stdair::Count_T & getActualTotalNumberOfEventsToBeGenerated () const
- const stdair::Count_T & getActualTotalNumberOfEventsToBeGenerated (const stdair::EventType::EN_EventType &) const
- const stdair::ProgressStatus & getStatus () const
- const stdair::ProgressStatus & getStatus (const stdair::EventType::EN_EventType &) const
- std::string describeKey () const
- std::string list () const
- std::string list (const stdair::EventType::EN_EventType &) const

- std::string jsonHandler (const stdair::JSONString &) const
- std::string jsonExportEventQueue (const stdair::EventType::EN_EventType &=stdair::EventType::LAST_VALUE) const
- std::string jsonExportEvent (const stdair::EventStruct &) const

24.14.1 Detailed Description

class holding the services related to Travel Demand Generation.

Definition at line 32 of file SEVMGR_Service.hpp.

24.14.2 Constructor & Destructor Documentation

24.14.2.1 SEVMGR::SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams & *iLogParams*, const stdair::BasDBParams & *iDBParams*)

Constructor. The initSevmgrService() 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 43 of file SEVMGR_Service.cpp.

24.14.2.2 SEVMGR::SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams & *iLogParams*)

Constructor.

The initSevmgrService() 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 64 of file SEVMGR_Service.cpp.

24.14.2.3 SEVMGR::SEVMGR_Service::SEVMGR_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_Service_ptr*)

Constructor.

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

Moreover, as no reference on any output stream is given, neither any database access parameter 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 SEVMGR_Service is itself being initialised by another library service such as TVLSIM_Service).

Parameters:

stdair::STDAIR_ServicePtr_T Handler on the STDAIR_Service.

Definition at line 85 of file SEVMGR_Service.cpp.

24.14.2.4 SEVMGR::SEVMGR_Service::∼SEVMGR_Service ()

Destructor.

Definition at line 101 of file SEVMGR_Service.cpp.

24.14.3 Member Function Documentation

24.14.3.1 void SEVMGR::SEVMGR_Service::buildSampleQueue ()

Build a sample event queue.

Definition at line 175 of file SEVMGR_Service.cpp.

Referenced by main().

24.14.3.2 stdair::BookingRequestStruct SEVMGR::SEVMGR_-Service::buildSampleBookingRequest (const bool *isForCRS* = `false`)

Build a sample booking request structure.

As of now (March 2011), the sample booking request is made of the following parameters:

- Return trip (inbound): LHR-SYD (POS: LHR, Channel: DN),
- Departing 10-JUN-2011 around 8:00, staying 7 days
- Requested on 15-MAY-2011 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

As of now (March 2011), the CRS-related booking request is made of the following parameters:

- Return trip (inbound): SIN-BKK (POS: SIN, Channel: IN),
- Departing 30-JAN-2010 around 10:00, staying 7 days
- Requested on 22-JAN-2010 at 10:00
- Economy cabin, 3 persons, FF member

- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

See also:

stdair::CmdBomManager for more details.

Parameters:

const bool isForCRS Whether the sample booking request is for CRS.

Returns:

BookingRequestStruct& Sample booking request structure.

Definition at line 200 of file SEVMGR_Service.cpp.

24.14.3.3 stdair::ProgressStatusSet SEVMGR::SEVMGR_Service::popEvent (stdair::EventStruct & *iEventStruct*) const

Pop the next coming (in time) event, and remove it from the event queue.

- The next coming (in time) event corresponds to the event having the earliest date-time stamp. In other words, it is the first/front element of the event queue.
- That (first) event/element is then removed from the event queue
- The progress status is updated for the corresponding event generator.

Returns:

stdair::EventStruct A copy of the event structure, which comes first in time from within the event queue.

Definition at line 398 of file SEVMGR_Service.cpp.

Referenced by main().

24.14.3.4 void SEVMGR::SEVMGR_Service::run (stdair::EventStruct & *iEventStruct*) const

Played all events and stopped when the first break point was encountered.

Returns:

stdair::EventStruct A copy of the break point which came first in time within the event queue. If no break point was encountered, return a copy of the last event within the event queue.

Definition at line 417 of file SEVMGR_Service.cpp.

24.14.3.5 bool SEVMGR::SEVMGR_Service::select (stdair::EventStruct & *iEventStruct*, const stdair::DateTime_T & *iEventDateTime*) const

Selected the event with the given date time, if such event existed.

Returns:

stdair::EventStruct A copy of the event with the given date time. If no event with the given DateTime was encountered, no copy are returned.

Parameters:

const stdair::DateTime_T Date time of the event to be returned.

Returns:

bool States whether an event with the given date time had been encountered and thus returned.

/Note All events occuring before the selected one are played. Thus, the copy returned is the copy of the current first event of the queue.

Definition at line 437 of file SEVMGR_Service.cpp.

24.14.3.6 template<class EventGenerator > void SEVMGR::SEVMGR_Service::addEventGenerator (EventGenerator & *iEventGenerator*) const `[inline]`

Add an event generator to the map holding the children of the queue. Be careful, this method is not implemented: its implementation is left to the appelant according the EventGenerator type.

Note:

An instance of implementation of that method can be found in the TraDemGen service.

24.14.3.7 void SEVMGR::SEVMGR_Service::addEvent (stdair::EventStruct & *iEventStruct*) const

Add an event to the queue.

Definition at line 596 of file SEVMGR_Service.cpp.

24.14.3.8 void SEVMGR::SEVMGR_Service::reset () const

Reset the context of the event generators for another event generation without having to reparse the demand input file.

Definition at line 561 of file SEVMGR_Service.cpp.

24.14.3.9 void SEVMGR::SEVMGR_Service::updateStatus (const stdair::EventType::EN_EventType & *iEventType*, const stdair::Count_T & *iEventCount*) const

Update the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Parameters:

const stdair::EventType::EN_EventType& Type of the events for which the actual total count is updated.

Returns:

const stdair::Count_T& Expected Actual count of such events already generated

Definition at line 458 of file SEVMGR_Service.cpp.

24.14.3.10 void SEVMGR::SEVMGR_Service::addStatus (const stdair::EventType::EN_-EventType & *iEventType*, const stdair::Count_T & *iEventCount*) const

Initialise the progress statuses for the given event type (e.g., request, snapshot).

Parameters:

const stdair::EventType::EN_EventType& Type of the events for which the actual total count is updated.

Returns:

const stdair::Count_T& Expected Actual count of such events already generated

Definition at line 478 of file SEVMGR_Service.cpp.

24.14.3.11 bool SEVMGR::SEVMGR_Service::isQueueDone () const

States whether the event queue has reached the end.

For now, that method states whether the event queue is empty.

Definition at line 497 of file SEVMGR_Service.cpp.

Referenced by main().

24.14.3.12 bool SEVMGR::SEVMGR_Service::hasProgressStatus (const stdair::EventType::EN_EventType & *iEventType*) const

Check if the event queue has already a progress status for the given event type

Definition at line 519 of file SEVMGR_Service.cpp.

24.14.3.13 EventQueue & SEVMGR::SEVMGR_Service::getEventQueue () const

Definition at line 580 of file SEVMGR_Service.cpp.

24.14.3.14 const stdair::Count_T & SEVMGR::SEVMGR_Service::getQueueSize () const

Get the size of the queue.

Definition at line 542 of file SEVMGR_Service.cpp.

24.14.3.15 template<class EventGenerator , class Key > EventGenerator& SEVMGR::SEVMGR_Service::getEventGenerator (const Key & *iKey*) const `[inline]`

Extract an event generator from the map holding the children of the queue. Be careful, this method is not implemented: its implementation is left to the appelant according the EventGenerator type.

Note:

An instance of implementation of that method can be found in the TraDemGen service.

24.14.3.16 template<class EventGenerator , class Key > bool SEVMGR::SEVMGR_-Service::hasEventGenerator (const Key & *iKey*) const `[inline]`

Check whether the event generator object with the given key exists.

Be careful, this method is not implemented: its implementation is left to the appelant according the Event-Generator type.

Note:

An instance of implementation of that method can be found in the TraDemGen service.

24.14.3.17 template<class EventGenerator > const std::list<EventGenerator*> SEVMGR::SEVMGR_Service::getEventGeneratorList () const `[inline]`

Extract the event generator list from the map holding the children of the queue. Be careful, this method is not implemented: its implementation is left to the appelant according the EventGenerator type.

Note:

An instance of implementation of that method can be found in the TraDemGen service.

24.14.3.18 template<class EventGenerator > bool SEVMGR::SEVMGR_-Service::hasEventGeneratorList () const `[inline]`

Check whether there are event generator objects.

Be careful, this method is not implemented: its implementation is left to the appelant according the Event-Generator type.

Note:

An instance of implementation of that method can be found in the TraDemGen service.

24.14.3.19 const stdair::Count_T & SEVMGR::SEVMGR_-Service::getExpectedTotalNumberOfEventsToBeGenerated () const

Get the expected number of events to be generated.

The getExpectedTotalNbOfEvents() method is called on the underlying EventQueue object, which keeps track of that number.

Note:

That number usually corresponds to an expectation (i.e., the mean value of a random distribution), and may not be accurate. The actual number will be known after calling the generateFirstEvents() method for each event type (e.g., booking request, optimisation notification, etc).

Returns:

const Count_T& Expected number of events to be generated.

Definition at line 616 of file SEVMGR_Service.cpp.

Referenced by getExpectedTotalNumberOfEventsToBeGenerated().

24.14.3.20 const stdair::Count_T & SEVMGR::SEVMGR_-Service::getExpectedTotalNumberOfEventsToBeGenerated (const stdair::EventType::EN_EventType & *iEventType*) const

Get the expected number of events to be generated for the given event type.

The getExpectedTotalNbOfEvents() method is called on the underlying EventQueue object, which keeps track of that number.

Note:

That number usually corresponds to an expectation (i.e., the mean value of a random distribution), and may not be accurate. The actual number will be known after calling the generateFirstEvents() method for each event type (e.g., booking request, optimisation notification, etc).

Parameters:

const EventType_T& Event type for which the number is calculated.

Returns:

const Count_T& Expected number of events to be generated.

Definition at line 636 of file SEVMGR_Service.cpp.

References getExpectedTotalNumberOfEventsToBeGenerated().

24.14.3.21 const stdair::Count_T & SEVMGR::SEVMGR_-Service::getActualTotalNumberOfEventsToBeGenerated () const

Get the actual number of events to be generated for all the event generators.

The getActualTotalNbOfEvents() method is called on the underlying EventQueue object, which keeps track of that number.

Note:

That number is being known after calling the generateFirstEvents() method.

Returns:

const Count_T& Expected number of events to be generated.

Definition at line 657 of file SEVMGR_Service.cpp.

Referenced by getActualTotalNumberOfEventsToBeGenerated().

24.14.3.22 const stdair::Count_T & SEVMGR::SEVMGR_Service::getActualTotalNumberOfEventsToBeGenerated (const stdair::EventType::EN_EventType & *iEventType*) const

Get the expected number of events to be generated for the given event type.

The getExpectedTotalNbOfEvents() method is called on the underlying EventQueue object, which keeps track of that number.

Note:

That number usually corresponds to an expectation (i.e., the mean value of a random distribution), and may not be accurate. The actual number will be known after calling the generateFirstEvents() method for each event type (e.g., booking request, optimisation notification, etc).

Parameters:

const EventType_T& Event type for which the number is calculated.

Returns:

const Count_T& Expected number of events to be generated.

Definition at line 678 of file SEVMGR_Service.cpp.

References getActualTotalNumberOfEventsToBeGenerated().

24.14.3.23 const stdair::ProgressStatus & SEVMGR::SEVMGR_Service::getStatus () const

Get the overall progress status (for the whole event queue).

Definition at line 715 of file SEVMGR_Service.cpp.

Referenced by getStatus().

24.14.3.24 const stdair::ProgressStatus & SEVMGR::SEVMGR_Service::getStatus (const stdair::EventType::EN_EventType & *iEventType*) const

Get the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 736 of file SEVMGR_Service.cpp.

References getStatus().

24.14.3.25 std::string SEVMGR::SEVMGR_Service::describeKey () const

Display (dump in the returned string) the key of the event queue.

Returns:

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

Definition at line 224 of file SEVMGR_Service.cpp.

24.14.3.26 std::string SEVMGR::SEVMGR_Service::list () const

Display (dump in the returned string) the event list of the event queue.

Returns:

std::string Output string in which the events are logged/dumped.

Definition at line 243 of file SEVMGR_Service.cpp.

Referenced by list().

24.14.3.27 std::string SEVMGR::SEVMGR_Service::list (const stdair::EventType::EN_-EventType & *iEventType*) const

Display (dump in the returned string) the event list for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Parameters:

const EventType_T& Event type for which the events are displayed

Returns:

std::string Output string in which the events are logged/dumped.

Definition at line 263 of file SEVMGR_Service.cpp.

References list().

24.14.3.28 std::string SEVMGR::SEVMGR_Service::jsonHandler (const stdair::JSONString & *iJSONString*) const

Dispatch the JSon command string to the corresponding service.

Parameters:

const stdair::JSONString& Input string which contained the JSon command string.

Returns:

std::string Output string in which the asking objects are logged/dumped with a JSon format.

Definition at line 283 of file SEVMGR_Service.cpp.

References jsonExportEventQueue().

24.14.3.29 std::string SEVMGR::SEVMGR_Service::jsonExportEventQueue (const stdair::EventType::EN_EventType & *iEventType* = `stdair::EventType::LAST_VALUE`) const

Dump in the returned string and in JSON format the whole list of events queue.

Definition at line 342 of file SEVMGR_Service.cpp.

Referenced by jsonHandler().

24.14.3.30 std::string SEVMGR::SEVMGR_Service::jsonExportEvent (const stdair::EventStruct & *iEvent*) const

Dump in the returned string and in JSON format the given event.

Definition at line 372 of file SEVMGR_Service.cpp.

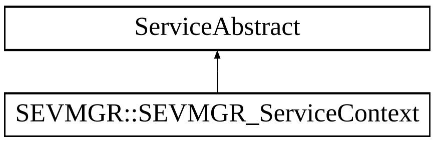
The documentation for this class was generated from the following files:

- sevmgr/SEVMGR_Service.hpp
- sevmgr/service/SEVMGR_Service.cpp

24.15 SEVMGR::SEVMGR_ServiceContext Class Reference

Class holding the context of the Sevmgr services.

`#include <sevmgr/service/SEVMGR_ServiceContext.hpp>`Inheritance diagram for SEVMGR::SEVMGR_ServiceContext::



Friends

- class SEVMGR_Service
- class FacSEVMGRServiceContext

24.15.1 Detailed Description

Class holding the context of the Sevmgr services.

Definition at line 30 of file SEVMGR_ServiceContext.hpp.

24.15.2 Friends And Related Function Documentation

24.15.2.1 friend class SEVMGR_Service `[friend]`

The SEVMGR_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 36 of file SEVMGR_ServiceContext.hpp.

24.15.2.2 friend class FacSEVMGRServiceContext `[friend]`

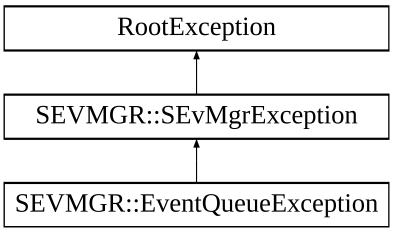
Definition at line 37 of file SEVMGR_ServiceContext.hpp.

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

- sevmgr/service/SEVMGR_ServiceContext.hpp
- sevmgr/service/SEVMGR_ServiceContext.cpp

24.16 SEVMGR::SEvMgrException Class Reference

`#include <sevmgr/SEVMGR_Exceptions.hpp>`Inheritance diagram for SEVMGR::SEvMgrException::



Public Member Functions

- SEvMgrException (const std::string &iWhat)

24.16.1 Detailed Description

Root exception for the Sevmgr component

Definition at line 18 of file SEVMGR_Exceptions.hpp.

24.16.2 Constructor & Destructor Documentation

24.16.2.1 SEVMGR::SEvMgrException::SEvMgrException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 23 of file SEVMGR_Exceptions.hpp.

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

- sevmgr/SEVMGR_Exceptions.hpp

25 File Documentation

25.1 doc/local/authors.doc File Reference

25.2 doc/local/codingrules.doc File Reference

25.3 doc/local/copyright.doc File Reference

25.4 doc/local/documentation.doc File Reference

25.5 doc/local/features.doc File Reference

25.6 doc/local/help_wanted.doc File Reference

25.7 doc/local/howto_release.doc File Reference

25.8 doc/local/index.doc File Reference

25.9 doc/local/installation.doc File Reference

25.10 doc/local/linking.doc File Reference

25.11 doc/local/test.doc File Reference

25.12 doc/local/users_guide.doc File Reference

25.13 doc/local/verification.doc File Reference

25.14 doc/tutorial/tutorial.doc File Reference

25.15 sevmgr/basic/BasConst.cpp File Reference

```
#include <stdair/basic/BasConst_General.hpp>
#include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>
```

Namespaces

- namespace SEVMGR

Functions

- const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID ("EQ01")

25.16 BasConst.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/basic/BasConst_General.hpp>
// Sevmgr
#include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>

namespace SEVMGR {

  // const std::string DEFAULT_SEVMGR_SERVICE_NAME = "sevmgr";

  const EventQueueID_T DEFAULT_EVENT_QUEUE_ID ("EQ01");

}
```

25.17 sevmgr/basic/BasConst_EventQueueManager.hpp File Reference

```
#include <string>
#include <sevmgr/SEVMGR_Types.hpp>
```

Namespaces

- namespace SEVMGR

Variables

- const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID

25.18 BasConst_EventQueueManager.hpp

```
#ifndef __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP
#define __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
//SEvMgr
#include <sevmgr/SEVMGR_Types.hpp>

namespace SEVMGR {

  extern const EventQueueID_T DEFAULT_EVENT_QUEUE_ID;

}
#endif // __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP
```

25.19 sevmgr/basic/BasConst_SEVMGR_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace SEVMGR

25.20 BasConst_SEVMGR_Service.hpp

```
#ifndef __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP
#define __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
#include <string>

namespace SEVMGR {

  // extern const std::string DEFAULT_SEVMGR_SERVICE_NAME;

}
#endif // __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP
```

25.21 sevmgr/basic/BasParserTypes.hpp File Reference

```
#include <string>
#include <boost/spirit/home/classic/core.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>
```

Namespaces

- namespace SEVMGR

Typedefs

- typedef char SEVMGR::char_t
- typedef boost::spirit::classic::file_iterator< char_t > SEVMGR::iterator_t
- typedef boost::spirit::classic::scanner< iterator_t > SEVMGR::scanner_t
- typedef boost::spirit::classic::rule< scanner_t > SEVMGR::rule_t
- typedef boost::spirit::classic::int_parser< unsigned int, 10, 1, 1 > SEVMGR::int1_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 2, 2 > SEVMGR::uint2_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 2 > SEVMGR::uint1_2_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 3 > SEVMGR::uint1_3_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 4, 4 > SEVMGR::uint4_p_t
- typedef boost::spirit::classic::uint_parser< unsigned int, 10, 1, 4 > SEVMGR::uint1_4_p_t
- typedef boost::spirit::classic::chset< char_t > SEVMGR::chset_t
- typedef boost::spirit::classic::impl::loop_traits< chset_t, unsigned int, unsigned int >::type SEVMGR::repeat_p_t
- typedef boost::spirit::classic::bounded< uint2_p_t, unsigned int > SEVMGR::bounded2_p_t
- typedef boost::spirit::classic::bounded< uint1_2_p_t, unsigned int > SEVMGR::bounded1_2_p_t
- typedef boost::spirit::classic::bounded< uint1_3_p_t, unsigned int > SEVMGR::bounded1_3_p_t
- typedef boost::spirit::classic::bounded< uint4_p_t, unsigned int > SEVMGR::bounded4_p_t
- typedef boost::spirit::classic::bounded< uint1_4_p_t, unsigned int > SEVMGR::bounded1_4_p_t

25.22 BasParserTypes.hpp

```
#ifndef __SEVMGR_BAS_BASCOMPARSERTYPES_HPP
#define __SEVMGR_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 SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  //
  //  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, 1, 2>
uint1_2_p_t;

  typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3>
uint1_3_p_t;

  typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t;

  typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
uint1_4_p_t;

  typedef boost::spirit::classic::chset<char_t> chset_t;

  typedef boost::spirit::classic::impl::loop_traits<chset_t,
                                          unsigned int,
                                          unsigned int>::type repeat_p_t;

  typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t;
  typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int>
bounded1_2_p_t;
  typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int>
bounded1_3_p_t;
  typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t;
  typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
bounded1_4_p_t;
}
#endif // __SEVMGR_BAS_BASCOMPARSERTYPES_HPP
```

25.23 sevmgr/batches/sevmgr_demo.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <fstream>
#include <vector>
#include <list>
#include <string>
#include <boost/program_options.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/config/sevmgr-paths.hpp>
```

Functions

- const stdair::Filename_T K_SEVMGR_DEFAULT_LOG_FILENAME ("sevmgr_demo.log")
- int readConfiguration (int argc, char ∗argv[ ], stdair::Filename_T &ioLogFilename)
- int main (int argc, char ∗argv[ ])

Variables

- const int K_SEVMGR_EARLY_RETURN_STATUS = 99

25.23.1 Function Documentation

25.23.1.1 const stdair::Filename_T K_SEVMGR_DEFAULT_LOG_FILENAME ("sevmgr_demo.log")

Default name and location for the log file.

Referenced by readConfiguration().

25.23.1.2 int readConfiguration (int *argc*, char ∗ *argv*[ ], stdair::Filename_T & *ioLogFilename*)

Read and parse the command line options.

Definition at line 37 of file sevmgr_demo.cpp.

References K_SEVMGR_DEFAULT_LOG_FILENAME(), K_SEVMGR_EARLY_RETURN_STATUS, PACKAGE_NAME, PACKAGE_VERSION, and PREFIXDIR.

Referenced by main().

25.23.1.3 int main (int *argc*, char ∗ *argv*[ ])

Definition at line 111 of file sevmgr_demo.cpp.

References SEVMGR::SEVMGR_Service::buildSampleQueue(), SEVMGR::SEVMGR_-Service::isQueueDone(), K_SEVMGR_EARLY_RETURN_STATUS, SEVMGR::SEVMGR_-Service::popEvent(), and readConfiguration().

25.23.2 Variable Documentation

25.23.2.1 const int K_SEVMGR_EARLY_RETURN_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 32 of file sevmgr_demo.cpp.

Referenced by main(), and readConfiguration().

25.24 sevmgr_demo.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
#include <fstream>
#include <vector>
#include <list>
#include <string>
//  //// Boost (Extended STL) ////
// Boost Program Options
#include <boost/program_options.hpp>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/bom/EventStruct.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/config/sevmgr-paths.hpp>

// //////// Constants //////
const stdair::Filename_T K_SEVMGR_DEFAULT_LOG_FILENAME ("sevmgr_demo.log");

const int K_SEVMGR_EARLY_RETURN_STATUS = 99;


// ///////// Parsing of Options & Configuration /////////
int readConfiguration (int argc, char* argv[],
                       stdair::Filename_T& ioLogFilename) {

  // 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()
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(
K_SEVMGR_DEFAULT_LOG_FILENAME),
     "Filepath for the logs")
    ;

  // 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 ("sevmgr.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_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("version")) {
    std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
    return K_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("prefix")) {
    std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
    return K_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}


// ////////////// M A I N ////////////////
int main (int argc, char* argv[]) {

  // Output log File
  stdair::Filename_T lLogFilename;

  // Call the command-line option parser
  const int lOptionParserStatus = readConfiguration (argc, argv, lLogFilename);

  if (lOptionParserStatus == K_SEVMGR_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();

  // Set up the log parameters
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);

  SEVMGR::SEVMGR_Service sevmgrService (lLogParams);

  // Build the default sample queue.
  STDAIR_LOG_DEBUG ("Build the default sample queue.");
  sevmgrService.buildSampleQueue();

  stdair::Count_T idx = 1;
  while (sevmgrService.isQueueDone() == false) {

    // Pop the next event out of the event queue
    stdair::EventStruct lEventStruct;
    const stdair::ProgressStatusSet lPPS =
      sevmgrService.popEvent (lEventStruct);

    // DEBUG
    STDAIR_LOG_DEBUG ("Poped event "<< idx << ": '"
                      << lEventStruct.describe() << "'.");
    STDAIR_LOG_DEBUG ("Progresss status: " << lPPS.describe());

    // Iterate
    ++idx;
  }

  // DEBUG
  STDAIR_LOG_DEBUG ("End of the simulation");

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

  /*
    Note: as that program is not intended to be run on a server in
    production, it is better not to catch the exceptions. When it
    happens (that an exception is throwned), that way we get the
    call stack.
  */

  return 0;
}
```

25.25 sevmgr/bom/BomJSONExport.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/bom/BomJSONExport.hpp>
```

Namespaces

- namespace bpt
- namespace SEVMGR

Typedefs

- typedef char bpt::ptree

25.26 BomJSONExport.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <ostream>
#if BOOST_VERSION >= 104100
// Boost Property Tree
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/regex.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/STDAIR_Service.hpp>
#include <stdair/bom/EventStruct.hpp>
// SEVMGR
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/bom/BomJSONExport.hpp>

#if BOOST_VERSION >= 104100
namespace bpt = boost::property_tree;
#else // BOOST_VERSION >= 104100
namespace bpt {
  typedef char ptree;
}
#endif // BOOST_VERSION >= 104100

namespace SEVMGR {

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::
  jsonExportEventQueue (stdair::STDAIR_ServicePtr_T& ioSTDAIR_ServicePtr,
                        std::ostream& oStream,
                        const EventQueue& iEventQueue,
                        const stdair::EventType::EN_EventType& iEventType) {

    // Retrieve the event list
    const stdair::EventList_T& lEventList = iEventQueue.getEventList();

#if BOOST_VERSION >= 104100
    // Create empty property tree objects
    bpt::ptree ptEvents;
    bpt::ptree pt;

    // Browse the events
    for (stdair::EventList_T::const_iterator itEvent = lEventList.begin();
         itEvent != lEventList.end(); ++itEvent) {
      const stdair::EventStruct& lEvent = itEvent->second;
      const stdair::EventType::EN_EventType& lEventType =
        lEvent.getEventType();

      const bool isEventTypeLastValue =
        (iEventType == stdair::EventType::LAST_VALUE);
      if (lEventType == iEventType || isEventTypeLastValue == true) {

        // Delegate the JSON export to the dedicated service
        const std::string lCurrentEvent =
          ioSTDAIR_ServicePtr->jsonExportEventObject (lEvent);

        // Load the JSON formatted string into the property tree.
        // If reading fails (cannot open stream, parse error), an
        // exception is thrown.
        if (lCurrentEvent.empty () == false) {
          bpt::ptree ptCurrentEvent;
          std::istringstream lStrCurrentEvent(lCurrentEvent);
          read_json (lStrCurrentEvent, ptCurrentEvent);

          // Put the current inventory tree in the events array
          ptEvents.push_back(std::make_pair("", ptCurrentEvent));
        }
      }
    }

    // Store the events array tree into the global tree
    pt.add_child ("events", ptEvents);

    // Write the property tree into the JSON stream.
    write_json (oStream, pt);

#endif // BOOST_VERSION >= 104100
  }

}
```

25.27 sevmgr/bom/BomJSONExport.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/EventTypes.hpp>
```

Classes

- class SEVMGR::BomJSONExport

  Utility class to export StdAir objects in a JSON format.

Namespaces

- namespace bpt
- namespace SEVMGR

25.28 BomJSONExport.hpp

```
#ifndef __SEVMGR_BOM_BOMJSONEXPORT_HPP
#define __SEVMGR_BOM_BOMJSONEXPORT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// Boost Property Tree
#if BOOST_VERSION >= 104100
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/EventTypes.hpp>

#if BOOST_VERSION >= 104100
  namespace bpt = boost::property_tree;
#else // BOOST_VERSION >= 104100
  namespace bpt {
    typedef char ptree;
  }
#endif // BOOST_VERSION >= 104100

namespace SEVMGR {

  class EventQueue;

  class BomJSONExport {
  public:
    // //////////////// Export support methods ////////////////

    static void jsonExportEventQueue (stdair::STDAIR_ServicePtr_T&,
                                      std::ostream&,
                                      const EventQueue&,
                                      const stdair::EventType::EN_EventType&);

  };

}
#endif // __SEVMGR_BOM_BOMJSONEXPORT_HPP
```

25.29 sevmgr/bom/EventQueue.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>
#include <sevmgr/bom/EventQueue.hpp>
```

Namespaces

- namespace SEVMGR

25.30 EventQueue.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/service/Logger.hpp>
// SEvMgr
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>
#include <sevmgr/bom/EventQueue.hpp>

namespace SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  EventQueue::EventQueue()
    : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
      _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
                       stdair::DEFAULT_PROGRESS_STATUS) {
  }

  // //////////////////////////////////////////////////////////////////////
  EventQueue::EventQueue (const Key_T& iKey)
    : _key (iKey), _parent (NULL),
      _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
                       stdair::DEFAULT_PROGRESS_STATUS) {
  }

  // //////////////////////////////////////////////////////////////////////
  EventQueue::EventQueue (const EventQueue& iEventQueue)
    : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
      _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
                       stdair::DEFAULT_PROGRESS_STATUS) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  EventQueue::~EventQueue() {
    _eventList.clear();
  }

  // //////////////////////////////////////////////////////////////////////
  std::string EventQueue::toString() const {
    std::ostringstream oStr;
    oStr << "(" << _eventList.size() << ") "
         << _progressStatus.getCurrentNb() << "/{"
         << _progressStatus.getExpectedNb() << ","
         << _progressStatus.getActualNb() << "}";
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  std::string EventQueue::display() const {
    std::ostringstream oStr;

    oStr << toString();

    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  std::string EventQueue::list () const {
    std::ostringstream oStr;
    oStr << describeKey () << std::endl;
    oStr << toString() << std::endl;

    // Browse the events
    for (stdair::EventList_T::const_iterator itEvent = _eventList.begin();
         itEvent != _eventList.end(); ++itEvent) {
      const stdair::EventStruct& lEvent = itEvent->second;

      oStr << lEvent.describe();
    }

    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  std::string EventQueue::
  list (const stdair::EventType::EN_EventType& iType) const {
      std::ostringstream oStr;
      oStr << describeKey () << std::endl;
      oStr << toString() << std::endl;
      oStr << "List " << stdair::EventType::getLabel(iType)
           << " events:" << std::endl;

      // Browse the events
      for (stdair::EventList_T::const_iterator itEvent = _eventList.begin();
           itEvent != _eventList.end(); ++itEvent) {
        const stdair::EventStruct& lEvent = itEvent->second;

        if (lEvent.getEventType() == iType) {
          oStr << lEvent.describe();
        }
      }
      return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::Count_T EventQueue::getQueueSize () const {
    return _eventList.size();
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueue::isQueueEmpty () const {
    return _eventList.empty();
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueue::isQueueDone () const {
    const bool isQueueEmpty = _eventList.empty();
    return isQueueEmpty;
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueue::reset () {
    // Reset only the current number of events, not the expected one
    _progressStatus.reset();

    // Empty the list of events
    _eventList.clear();

    // Reset the progress statuses for all the event types
    for (ProgressStatusMap_T::iterator itProgressStatus =
           _progressStatusMap.begin();
         itProgressStatus != _progressStatusMap.end(); ++itProgressStatus) {
      stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
      lProgressStatus.reset();
    }
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueue::
  hasProgressStatus (const stdair::EventType::EN_EventType& iType) const {

    bool hasProgressStatus = true;

    // Retrieve the ProgressStatus structure corresponding to the
    // given event type
    ProgressStatusMap_T::const_iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus == _progressStatusMap.end()) {
      //
      STDAIR_LOG_DEBUG ("No ProgressStatus structure can be retrieved in the "
                        << "EventQueue: " << display());

      hasProgressStatus = false;
    }

    return hasProgressStatus;
  }

  // //////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueue::
  getCurrentNbOfEvents (const stdair::EventType::EN_EventType& iType) const {

    // Retrieve the ProgressStatus structure corresponding to the
    // given event type
    ProgressStatusMap_T::const_iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus == _progressStatusMap.end()) {
      //
      STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
                        << "EventQueue: " << display());
      assert (false);
    }

    const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
    return lProgressStatus.getCurrentNb();
  }

  // //////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueue::
  getExpectedTotalNbOfEvents (const stdair::EventType::EN_EventType& iType) const
 {

    // Retrieve the ProgressStatus structure corresponding to the
    // given event type
    ProgressStatusMap_T::const_iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus == _progressStatusMap.end()) {
      std::ostringstream oStr;
      oStr << "No ProgressStatus structure can be retrieved in the EventQueue '"
           << display() << "'. The EventQueue should be initialised, e.g., by "
           << "calling a buildSampleBom() method.";
      //
      STDAIR_LOG_ERROR (oStr.str());
      throw EventQueueException (oStr.str());
    }

    const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
    return lProgressStatus.getExpectedNb();
  }

  // //////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueue::
  getActualTotalNbOfEvents (const stdair::EventType::EN_EventType& iType) const {

    // Retrieve the ProgressStatus structure corresponding to the
    // given event type
    ProgressStatusMap_T::const_iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus == _progressStatusMap.end()) {
      //
      STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
                        << "EventQueue: " << display());
      assert (false);
    }

    const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
    return lProgressStatus.getActualNb();
  }

  // ////////////////////////////////////////////////////////////////////
  void EventQueue::updateStatus (const stdair::EventType::EN_EventType& iType,
                                 const stdair::ProgressStatus& iProgressStatus) {

    // Retrieve, if existing, the ProgressStatus structure
    // corresponding to the given event type
    ProgressStatusMap_T::iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus == _progressStatusMap.end()) {
      const bool hasInsertBeenSuccessful =
        _progressStatusMap.insert (ProgressStatusMap_T::
                                   value_type (iType, iProgressStatus)).second;

      if (hasInsertBeenSuccessful == false) {
        STDAIR_LOG_ERROR ("No progress_status can be inserted "
                          << "for the following event type: "
                          << stdair::EventType::getLabel(iType)
                          << ". EventQueue: " << toString());
        throw stdair::EventException ("No progress_status can be inserted for the
 "
                                      "following event type: "
                                      + stdair::EventType::getLabel(iType)
                                      + ". EventQueue: " + toString());
      }

      return;
    }

    stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;

    // Update the progress status
    const stdair::Count_T& lCurrentNb = iProgressStatus.getCurrentNb();
    lProgressStatus.setCurrentNb (lCurrentNb);

    const stdair::Count_T& lExpectedNb = iProgressStatus.getExpectedNb();
    lProgressStatus.setExpectedNb(lProgressStatus.getExpectedNb() + lExpectedNb);

    const stdair::Count_T& lActualNb = iProgressStatus.getActualNb();
    lProgressStatus.setActualNb (lProgressStatus.getActualNb() + lActualNb);
  }

  // ////////////////////////////////////////////////////////////////////
  void EventQueue::
  addStatus (const stdair::EventType::EN_EventType& iType,
             const stdair::NbOfEvents_T& iExpectedTotalNbOfEvents) {

    // Initialise the progress status object
    const stdair::Count_T lExpectedTotalNbOfEventsInt =
      static_cast<const stdair::Count_T> (std::floor (iExpectedTotalNbOfEvents));
    const stdair::ProgressStatus lProgressStatus (lExpectedTotalNbOfEventsInt);

    // Update the progress status for the given event type
    updateStatus (iType, lProgressStatus);

    // Update the overall progress status
    const stdair::Count_T lExpectedNb =
      static_cast<const stdair::Count_T> (_progressStatus.getExpectedNb()
                                  + iExpectedTotalNbOfEvents);
    _progressStatus.setExpectedNb (lExpectedNb);

    const stdair::Count_T lActualNb =
      static_cast<const stdair::Count_T> (_progressStatus.getActualNb()
                                  + iExpectedTotalNbOfEvents);
    _progressStatus.setActualNb (lActualNb);

  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueue::updateStatus (const stdair::EventType::EN_EventType& iType,
                                 const stdair::NbOfEvents_T& iActualNbOfEvents) {

    // Initialise the progress status object for the type key
    const stdair::Count_T lActualNbOfEventsInt =
      static_cast<const stdair::Count_T> (std::floor (iActualNbOfEvents));

    // Update the progress status for the corresponding content type key
    ProgressStatusMap_T::iterator itProgressStatus =
      _progressStatusMap.find (iType);
    if (itProgressStatus != _progressStatusMap.end()) {

      //
      stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;

      // Update the overall progress status
      const stdair::Count_T& lActualEventTypeNb = lProgressStatus.getActualNb();
      const stdair::Count_T lActualEventTypeNbInt =
        static_cast<const stdair::Count_T> (std::floor (lActualEventTypeNb));
      const stdair::Count_T& lActualTotalNb = _progressStatus.getActualNb();
      const stdair::Count_T lActualDeltaNb = lActualNbOfEventsInt - lActualEventTypeNbInt;
      _progressStatus.setActualNb (lActualTotalNb + lActualDeltaNb);

      // Update the progress status for the corresponding type key
      lProgressStatus.setActualNb (lActualNbOfEventsInt);
    }
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueue::setStatus (const stdair::EventType::EN_EventType& iType,
                              const stdair::ProgressStatus& iProgressStatus) {

    // Retrieve the ProgressStatus structure corresponding to the
    // given event type
    ProgressStatusMap_T::iterator itProgressStatus =
      _progressStatusMap.find (iType);
    // assert (itProgressStatus != _progressStatusMap.end());
    if (itProgressStatus != _progressStatusMap.end()) {
      // Update the ProgressStatus structure
      itProgressStatus->second = iProgressStatus;
    }
  }

  // //////////////////////////////////////////////////////////////////////
```

```
const stdair::ProgressStatus& EventQueue::
getStatus (const stdair::EventType::EN_EventType& iType) const {

  // Retrieve the ProgressStatus structure corresponding to the
  // given event type
  ProgressStatusMap_T::const_iterator itProgressStatus =
    _progressStatusMap.find (iType);
  if (itProgressStatus == _progressStatusMap.end()) {
    std::ostringstream oStr;
    oStr << "No ProgressStatus structure can be retrieved in the EventQueue '"

         << display() << "' for the following event type: "
         << stdair::EventType::getLabel(iType) << ".";
    //
    STDAIR_LOG_ERROR (oStr.str());
    throw EventQueueException (oStr.str());
  }
  assert(itProgressStatus != _progressStatusMap.end());

  const stdair::ProgressStatus& oProgressStatus = itProgressStatus->second;
  return oProgressStatus;
}

// ////////////////////////////////////////////////////////////////////////
stdair::ProgressPercentage_T EventQueue::
calculateProgress (const stdair::EventType::EN_EventType& iType) const {

  // Retrieve the ProgressStatus structure corresponding to the
  // given event type
  ProgressStatusMap_T::const_iterator itProgressStatus =
    _progressStatusMap.find (iType);
  if (itProgressStatus == _progressStatusMap.end()) {
    //
    STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
                      << "EventQueue: " << display());
    assert (false);
  }

  const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
  return lProgressStatus.progress();
}

// ////////////////////////////////////////////////////////////////////////
stdair::ProgressStatusSet EventQueue::popEvent (stdair::EventStruct& ioEventStr
uct) {

  if (_eventList.empty() == true) {
    std::ostringstream oStr;
    oStr << "The event queue '" << describeKey() << "' is empty. "
         << "No event can be popped.";
    //
    STDAIR_LOG_ERROR (oStr.str());
    throw EventQueueException (oStr.str());
  }

  // Get an iterator on the first event (sorted by date-time stamps)
  stdair::EventList_T::iterator itEvent = _eventList.begin();

  ioEventStruct = itEvent->second;
  // Retrieve the event type
  const stdair::EventType::EN_EventType& lEventType = ioEventStruct.getEventTyp
e();
  stdair::ProgressStatusSet oProgressStatusSet (lEventType);

  // Update the (current number part of the) overall progress status,
  // to account for the event that is being popped out of the event
  // queue.
```

```
    ++_progressStatus;

    // Remove the event, which has just been retrieved
    _eventList.erase (itEvent);


    // Retrieve the progress status specific to that event type
    stdair::ProgressStatus lEventTypeProgressStatus = getStatus (lEventType);

    // Increase the current number of events
    ++lEventTypeProgressStatus;

    // Store back the progress status
    setStatus (lEventType, lEventTypeProgressStatus);

    // Update the progress status of the progress status set, specific to
    // the event type.
    oProgressStatusSet.setTypeSpecificStatus (lEventTypeProgressStatus);

    // Update the overall progress status of the progress status set.
    oProgressStatusSet.setOverallStatus (_progressStatus);

    //
    return oProgressStatusSet;
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueue::addEvent (stdair::EventStruct& ioEventStruct) {
    bool insertionSucceeded =
      _eventList.insert (stdair::EventListElement_T (ioEventStruct.getEventTimeSt
amp(),
                                                     ioEventStruct)).second;

    const unsigned int idx = 0;
    while (insertionSucceeded == false && idx != 1e3) {
      // Increment the date-time stamp (expressed in milliseconds)
      ioEventStruct.incrementEventTimeStamp();

      // Retry to insert into the event queue
      insertionSucceeded =
        _eventList.insert (stdair::EventListElement_T (ioEventStruct.getEventTime
Stamp(),
                                                       ioEventStruct)).second;
    }
    assert (idx != 1e3);

    return insertionSucceeded;
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueue::hasEventDateTime (const stdair::DateTime_T& iDateTime) {

    bool hasSearchEventBeenSucessful = true;

    const stdair::Duration_T lDuration =
      iDateTime - stdair::DEFAULT_EVENT_OLDEST_DATETIME;
    const stdair::LongDuration_T lDateTimeStamp =
      lDuration.total_milliseconds();

    // Searches the container for an element with iDateTime as key
    stdair::EventList_T::iterator itEvent =
      _eventList.find (lDateTimeStamp);

    // An iterator to map::end means the specified key has not found in the
    // container.
    if (itEvent == _eventList.end()) {
      hasSearchEventBeenSucessful = false;
    }

    return hasSearchEventBeenSucessful;

  }

}
```

25.31 sevmgr/bom/EventQueue.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <sevmgr/bom/EventQueueKey.hpp>
#include <sevmgr/bom/EventQueueTypes.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class SEVMGR::EventQueue

  Class holding event structures.

Namespaces

- namespace stdair

  Forward declarations.

- namespace SEVMGR

25.32 EventQueue.hpp

```
#ifndef __SEVMGR_BOM_EVENTQUEUE_HPP
#define __SEVMGR_BOM_EVENTQUEUE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/EventTypes.hpp>
// SEvMgr
#include <sevmgr/bom/EventQueueKey.hpp>
#include <sevmgr/bom/EventQueueTypes.hpp>
#include <sevmgr/SEVMGR_Types.hpp>

namespace stdair {
  class FacBomManager;
  template <typename BOM> class FacBom;
}

namespace SEVMGR {

  class EventQueue : public stdair::BomAbstract {
    template <typename BOM> friend class stdair::FacBom;
    friend class stdair::FacBomManager;

  public:
    // ////////// Type definitions ////////////
    typedef EventQueueKey Key_T;


  public:
    // /////////// Getters ///////////////
    const Key_T& getKey () const {
      return _key;
    }

    BomAbstract* const getParent () const {
      return _parent;
    }

    const stdair::EventList_T& getEventList () const {
      return _eventList;
    }

    const stdair::HolderMap_T& getHolderMap () const {
      return _holderMap;
    }

    const stdair::ProgressStatus& getStatus () const {
      return _progressStatus;
    }
    const stdair::Count_T& getCurrentNbOfEvents () const {
      return _progressStatus.getCurrentNb();
    }
    const stdair::Count_T& getExpectedTotalNbOfEvents () const {
      return _progressStatus.getExpectedNb();
    }
    const stdair::Count_T& getActualTotalNbOfEvents () const {
      return _progressStatus.getActualNb();
    }

    const stdair::ProgressStatus& getStatus (const stdair::EventType::EN_EventType&) const;

    const stdair::Count_T& getCurrentNbOfEvents (const stdair::EventType::EN_EventType&) const;

    const stdair::Count_T& getExpectedTotalNbOfEvents (const stdair::EventType::EN_EventType&) const;

    const stdair::Count_T& getActualTotalNbOfEvents (const stdair::EventType::EN_EventType&) const;

    bool hasProgressStatus (const stdair::EventType::EN_EventType&) const;

  public:
    // /////////// Setters ///////////////
    void setStatus (const stdair::ProgressStatus& iProgressStatus) {
      _progressStatus = iProgressStatus;
    }
    void setStatus (const stdair::Count_T& iCurrentNbOfEvents,
                    const stdair::Count_T& iExpectedTotalNbOfEvents,
                    const stdair::Count_T& iActualTotalNbOfEvents) {
      _progressStatus.setCurrentNb (iCurrentNbOfEvents);
      _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
      _progressStatus.setActualNb (iActualTotalNbOfEvents);
    }
    void setStatus (const stdair::Count_T& iCurrentNbOfEvents,
                    const stdair::Count_T& iActualTotalNbOfEvents) {
      _progressStatus.setCurrentNb (iCurrentNbOfEvents);
      _progressStatus.setActualNb (iActualTotalNbOfEvents);
    }
    void setCurrentNbOfEvents (const stdair::Count_T& iCurrentNbOfEvents) {
      _progressStatus.setCurrentNb (iCurrentNbOfEvents);
    }
    void setExpectedTotalNbOfEvents (const stdair::Count_T& iExpectedTotalNbOfEvents) {
      _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
    }

    void setStatus (const stdair::EventType::EN_EventType& iType,
                    const stdair::ProgressStatus& iProgressStatus);


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString () const;

    std::string list () const;

    std::string list (const stdair::EventType::EN_EventType&) const;

    const std::string describeKey () const {
      return _key.toString();
    }

    /*
     * Display the full content of the event queue, with all its
     * event structure.
     *
     * That method can be very consuming (in time, CPU and memory)
     * when there are a lot of event structures (e.g., several hundreds
     * of thousands). Call it only for debug purposes.
    */
    std::string display () const;


  public:
    // ////////// Business methods /////////
    void reset ();

     stdair::ProgressStatusSet popEvent (stdair::EventStruct&);

    bool addEvent (stdair::EventStruct&);

    bool hasEventDateTime (const stdair::DateTime_T&);

    bool isQueueDone () const;

    void addStatus (const stdair::EventType::EN_EventType&,
                    const stdair::NbOfRequests_T& iExpectedTotalNbOfEvents);

    void updateStatus (const stdair::EventType::EN_EventType&,
                       const stdair::ProgressStatus& iProgressStatus);

    void updateStatus (const stdair::EventType::EN_EventType&,
                       const stdair::NbOfEvents_T& iActualTotalNbOfEvents);

    stdair::ProgressPercentage_T calculateProgress () const {
      return _progressStatus.progress();
    }

    stdair::ProgressPercentage_T calculateProgress (const stdair::EventType::EN_EventType&)const;


  public:
    // ////////// Debug methods /////////
    stdair::Count_T getQueueSize () const;

    bool isQueueEmpty () const;


  protected:
    // ////////// Constructors and destructors /////////
    EventQueue (const Key_T&);
    EventQueue (const EventQueue&);
    ~EventQueue ();
  private:
    EventQueue ();


  protected:
    // ////////// Attributes /////////
    Key_T _key;

    BomAbstract* _parent;

    stdair::HolderMap_T _holderMap;

    stdair::EventList_T _eventList;

    stdair::ProgressStatus _progressStatus;

    ProgressStatusMap_T _progressStatusMap;
  };

}
#endif // __SEVMGR_BOM_EVENTQUEUE_HPP
```

25.33 sevmgr/bom/EventQueueKey.cpp File Reference

```
#include <sstream>
#include <sevmgr/bom/EventQueueKey.hpp>
```

Namespaces

- namespace SEVMGR

25.34 EventQueueKey.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
// SEvMgr
#include <sevmgr/bom/EventQueueKey.hpp>

namespace SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  EventQueueKey::EventQueueKey (const EventQueueID_T& iEventQueueID)
    : _eventQueueID (iEventQueueID) {
  }
  // //////////////////////////////////////////////////////////////////////
  EventQueueKey::EventQueueKey (const EventQueueKey& iKey)
    : _eventQueueID (iKey._eventQueueID) {
  }

  // //////////////////////////////////////////////////////////////////////
  EventQueueKey::~EventQueueKey () {
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueueKey::toStream (std::ostream& ioOut) const {
    ioOut << "EventQueueKey: " << toString() << std::endl;
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueueKey::fromStream (std::istream& ioIn) {
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string EventQueueKey::toString() const {
    std::ostringstream oStr;
    oStr << _eventQueueID;
    return oStr.str();
  }

}
```

25.35 sevmgr/bom/EventQueueKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- struct SEVMGR::EventQueueKey

Namespaces

- namespace SEVMGR

25.36 EventQueueKey.hpp

```
#ifndef __SEVMGR_BOM_EVENTQUEUEKEY_HPP
#define __SEVMGR_BOM_EVENTQUEUEKEY_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
//SEvMgr
#include <sevmgr/SEVMGR_Types.hpp>

namespace SEVMGR {

  struct EventQueueKey : public stdair::KeyAbstract {

  private:
    // /////////// Default constructor //////////
    EventQueueKey () { };

  public:
    // /////////// Construction ///////////
    EventQueueKey (const EventQueueID_T&);
    EventQueueKey (const EventQueueKey&);
    ~EventQueueKey ();

    // /////////// Getters //////////
    const EventQueueID_T& getEventQueueID() const {
      return _eventQueueID;
    }

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  private:
    // //////////// Attributes ////////////////
    EventQueueID_T _eventQueueID;
  };

}
#endif // __SEVMGR_BOM_EVENTQUEUEKEY_HPP
```

25.37 sevmgr/bom/EventQueueTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace SEVMGR

Typedefs

- typedef std::list< EventQueue $*$ > SEVMGR::EventQueueList_T
- typedef std::map< const stdair::MapKey_T, EventQueue $*$ > SEVMGR::EventQueueMap_T

25.38 EventQueueTypes.hpp

```
// //////////////////////////////////////////////////////////////////////
#ifndef __SEVMGR_BOM_EVENTQUEUETYPES_HPP
#define __SEVMGR_BOM_EVENTQUEUETYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace SEVMGR {

  // Forward declarations.
  class EventQueue;

  typedef std::list<EventQueue*> EventQueueList_T;

  typedef std::map<const stdair::MapKey_T, EventQueue*> EventQueueMap_T;

}
#endif // __SEVMGR_BOM_EVENTQUEUETYPES_HPP
```

25.39 sevmgr/command/EventQueueManager.cpp File Reference

```
#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/command/EventQueueManager.hpp>
```

Namespaces

- namespace SEVMGR

25.40 EventQueueManager.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// Boost
#include <boost/make_shared.hpp>
// StdAir
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
// SEvMgr
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/command/EventQueueManager.hpp>

namespace SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  void EventQueueManager::
  buildSampleQueue (stdair::STDAIR_ServicePtr_T lSTDAIR_ServicePtr,
                    EventQueue& ioEventQueue) {

    // Total number of booking requests into the queue
    stdair::Count_T lNbOfBookingRequests (2);
    addStatus(ioEventQueue, stdair::EventType::BKG_REQ, lNbOfBookingRequests);

    // Create a shared pointer on a first booking request
    // Date of the request (15-MAY-2011)
    const stdair::BookingRequestStruct& lBookingRequest =
      buildSampleBookingRequest (lSTDAIR_ServicePtr);
    const stdair::BookingRequestPtr_T lBookingRequest_ptr =
      boost::make_shared<stdair::BookingRequestStruct> (lBookingRequest);

    // Create an event structure
    stdair::EventStruct lEventStruct (stdair::EventType::BKG_REQ,
                                      lBookingRequest_ptr);

    // Add the event into the queue
    addEvent(ioEventQueue, lEventStruct);

    // Create a second shared pointer on a second booking request
    // Date of the request (22-JAN-2010)
    const bool isForCRS = true;
    const stdair::BookingRequestStruct& lBookingRequestForCRS =
      buildSampleBookingRequest (lSTDAIR_ServicePtr, isForCRS);
    const stdair::BookingRequestPtr_T lBookingRequestForCRS_ptr =
      boost::make_shared<stdair::BookingRequestStruct> (lBookingRequestForCRS);

    // Create an event structure
    stdair::EventStruct lEventStructForCRS (stdair::EventType::BKG_REQ,
                                            lBookingRequestForCRS_ptr);

    // Add the event into the queue
    addEvent(ioEventQueue, lEventStructForCRS);

    // Total number of break points into the queue
    stdair::Count_T lNbOfBreakPoints (2);
    addStatus(ioEventQueue, stdair::EventType::BRK_PT, lNbOfBreakPoints);

    // Create a shared pointer on a second break point
    // Date of the break point (21-JAN-2010)
    const stdair::Date_T lBP1Date (2010, boost::gregorian::Jan, 21);
    // Time of the break point (00:00)
    const stdair::Duration_T lBP1Time (0, 0, 0);
    // Date-time of the break point  (made of the date and time above)
    const stdair::DateTime_T lBP1DateTime (lBP1Date, lBP1Time);
    const stdair::BreakPointPtr_T lBreakPoint1_ptr =
      boost::make_shared<stdair::BreakPointStruct> (lBP1DateTime);

    // Create an event structure
    stdair::EventStruct lEventBreakPoint1 (stdair::EventType::BRK_PT,
                                           lBreakPoint1_ptr);

    // Add the event into the queue
    addEvent(ioEventQueue, lEventBreakPoint1);

    // Create a shared pointer on a second break point
    // Date of the break point (14-MAY-2011)
    const stdair::Date_T lBP2Date (2011, boost::gregorian::May, 14);
    // Time of the break point (00:00)
    const stdair::Duration_T lBP2Time (0, 0, 0);
    // Date-time of the break point  (made of the date and time above)
    const stdair::DateTime_T lBP2DateTime (lBP2Date, lBP2Time);

    // TODO: understand why the form above does not work.
    const stdair::BreakPointPtr_T lBreakPoint2_ptr =
      boost::make_shared<stdair::BreakPointStruct> (lBP2DateTime);

    // Create an event structure
    stdair::EventStruct lEventBreakPoint2 (stdair::EventType::BRK_PT,
                                           lBreakPoint2_ptr);

    // Add the event into the queue
    addEvent(ioEventQueue, lEventBreakPoint2);

  }

  // //////////////////////////////////////////////////////////////////////
  stdair::BookingRequestStruct EventQueueManager::
  buildSampleBookingRequest(stdair::STDAIR_ServicePtr_T lSTDAIR_ServicePtr,
                            const bool isForCRS) {

    // Delegate the booking request building to the dedicated service
    stdair::BookingRequestStruct oBookingRequest =
      lSTDAIR_ServicePtr->buildSampleBookingRequest (isForCRS);

    return oBookingRequest;
  }


  // //////////////////////////////////////////////////////////////////////
  void EventQueueManager::reset (EventQueue& ioEventQueue) {

    ioEventQueue.reset();
  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueueManager::
  hasProgressStatus (const EventQueue& iEventQueue,
                     const stdair::EventType::EN_EventType& iEventType) {

    const bool hasProgressStatus = iEventQueue.hasProgressStatus(iEventType);

    //
    return hasProgressStatus;
  }
```

```

// //////////////////////////////////////////////////////////////////////
void EventQueueManager::addEvent (EventQueue& ioEventQueue,
                                  stdair::EventStruct& iEventStruct) {

  ioEventQueue.addEvent(iEventStruct);
}

// //////////////////////////////////////////////////////////////////////
const std::string EventQueueManager::
list (const EventQueue& iEventQueue) {

  const std::string& lEventListStr = iEventQueue.list();

  //
  return lEventListStr;
}

// //////////////////////////////////////////////////////////////////////
const std::string EventQueueManager::
list (const EventQueue& iEventQueue,
      const stdair::EventType::EN_EventType& iEventType) {

  const std::string& lEventListStr =
    iEventQueue.list(iEventType);

  //
  return lEventListStr;
}

// //////////////////////////////////////////////////////////////////////
const std::string EventQueueManager::
describeKey (const EventQueue& iEventQueue) {

  const std::string& lEventQueueKeyStr = iEventQueue.describeKey();

  //
  return lEventQueueKeyStr;
}

// //////////////////////////////////////////////////////////////////////
stdair::ProgressStatusSet EventQueueManager::
popEvent (EventQueue& ioEventQueue,
          stdair::EventStruct& iEventStruct) {

  try {
    const stdair::ProgressStatusSet& lProgressStatusSet
      = ioEventQueue.popEvent (iEventStruct);

    // DEBUG
    std::ostringstream oEventStr;
    oEventStr << "Poped event: '"
              << iEventStruct.describe() << "'.";
    STDAIR_LOG_DEBUG (oEventStr.str());

    //
    return lProgressStatusSet;

  } catch (EventQueueException& lEventQueueException) {
    // DEBUG
    std::ostringstream oErrorMessage;
    oErrorMessage << "The event queue is empty: no event can be popped out.";

    std::cerr << oErrorMessage.str() << std::endl;
    STDAIR_LOG_DEBUG(oErrorMessage.str());

  }
```

```
    //
    return stdair::ProgressStatusSet(stdair::EventType::BKG_REQ);
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueueManager::run (EventQueue& ioEventQueue,
                               stdair::EventStruct& iEventStruct) {

    // Default event type
    stdair::EventType::EN_EventType lEventType = stdair::EventType::BKG_REQ;

    // While no break point has been encountered, keep on extracting events
    while (ioEventQueue.isQueueDone() == false
           && lEventType != stdair::EventType::BRK_PT) {
      ioEventQueue.popEvent (iEventStruct);
      lEventType = iEventStruct.getEventType();

    }

  }

  // //////////////////////////////////////////////////////////////////////
  bool EventQueueManager::select (EventQueue& ioEventQueue,
                                  stdair::EventStruct& iEventStruct,
                                  const stdair::DateTime_T& iDateTime) {

    // Search if an event has the given key
    const bool hasResearchBeenSuccessful =
      ioEventQueue.hasEventDateTime (iDateTime);

    // If no event has the given hey, return
    if (hasResearchBeenSuccessful == false) {
      return hasResearchBeenSuccessful;
    }
    assert (hasResearchBeenSuccessful == true);

    // Default date time
    stdair::DateTime_T lDateTime = stdair::DEFAULT_EVENT_OLDEST_DATETIME;

    // While the event with the given key has not been retrieved, keep on
    // extracting events
    while (ioEventQueue.isQueueDone() == false
           && lDateTime != iDateTime) {
      ioEventQueue.popEvent (iEventStruct);
      lDateTime = iEventStruct.getEventTime ();

    }

    assert (lDateTime == iDateTime);
    return hasResearchBeenSuccessful;

  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueueManager::
  updateStatus (EventQueue& ioEventQueue,
                const stdair::EventType::EN_EventType& iEventType,
                const stdair::Count_T& iEventCount) {

    ioEventQueue.updateStatus (iEventType, iEventCount);
  }

  // //////////////////////////////////////////////////////////////////////
  void EventQueueManager::
  addStatus (EventQueue& ioEventQueue,
             const stdair::EventType::EN_EventType& iEventType,
               const stdair::Count_T& iEventCount) {

    ioEventQueue.addStatus (iEventType, iEventCount);
  }

  // ////////////////////////////////////////////////////////////////////
  bool EventQueueManager::
  isQueueDone (const EventQueue& iEventQueue) {

    const bool isQueueDone = iEventQueue.isQueueDone();

    //
    return isQueueDone;
  }

  // ////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueueManager::
  getQueueSize (const EventQueue& iEventQueue) {

    const stdair::Count_T& lQueueSize = iEventQueue.getQueueSize();

    //
    return lQueueSize;
  }

  // ////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueueManager::
  getExpectedTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue) {

    const stdair::Count_T& lExpectedTotalNumberOfEvents =
      ioEventQueue.getExpectedTotalNbOfEvents ();

    //
    return lExpectedTotalNumberOfEvents;
  }

  // ////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueueManager::
  getExpectedTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue,
                                               const stdair::EventType::EN_EventT
ype& iEventType) {

    const stdair::Count_T& lExpectedTotalNumberOfEvents =
      ioEventQueue.getExpectedTotalNbOfEvents (iEventType);

    //
    return lExpectedTotalNumberOfEvents;
  }

  // ////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueueManager::
  getActualTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue) {

    const stdair::Count_T& lActualTotalNumberOfEvents =
      ioEventQueue.getActualTotalNbOfEvents ();

    //
    return lActualTotalNumberOfEvents;

  }

  // ////////////////////////////////////////////////////////////////////
  const stdair::Count_T& EventQueueManager::
  getActualTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue,
                                             const stdair::EventType::EN_EventTyp
e& iEventType) {

    const stdair::Count_T& lActualTotalNumberOfEvents =
      ioEventQueue.getActualTotalNbOfEvents (iEventType);

    //
    return lActualTotalNumberOfEvents;

  }

  const stdair::ProgressStatus& EventQueueManager::
  getStatus (const EventQueue& iEventQueue,
             const stdair::EventType::EN_EventType& iEventType) {

    return iEventQueue.getStatus(iEventType);

  }

  const stdair::ProgressStatus&  EventQueueManager::
  getStatus (const EventQueue& iEventQueue) {

    return iEventQueue.getStatus();

  }

}
```

25.41 sevmgr/command/EventQueueManager.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <stdair/stdair_service_types.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class SEVMGR::EventQueueManager

  Utility class for Demand and DemandStream objects.

Namespaces

- namespace stdair

  Forward declarations.

- namespace SEVMGR

25.42 EventQueueManager.hpp

```
#ifndef __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP
#define __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <stdair/stdair_service_types.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Types.hpp>

// Forward declarations
namespace stdair {
  struct ProgressStatusSet;
}

namespace SEVMGR {

  // Forward declarations
  class EventQueue;

  class EventQueueManager : public stdair::CmdAbstract {
    friend class SEVMGR_Service;

  private:
    // //////// Business methodes //////////
    static void buildSampleQueue (stdair::STDAIR_ServicePtr_T,
                                  EventQueue&);

    static stdair::BookingRequestStruct buildSampleBookingRequest(stdair::STDAIR_ServicePtr_T,
                                                                  const bool isForCRS = false);

    static void reset (EventQueue&);

    static void addEvent (EventQueue&, stdair::EventStruct&);

    static const std::string describeKey (const EventQueue&);

    static const std::string list (const EventQueue&);

    static const std::string list (const EventQueue&,
                                   const stdair::EventType::EN_EventType&);

    static stdair::ProgressStatusSet popEvent (EventQueue&,
                                               stdair::EventStruct&);

    static void run (EventQueue&, stdair::EventStruct&);

    static bool select (EventQueue&, stdair::EventStruct&, const stdair::DateTime_T&);

    static void updateStatus (EventQueue&,
                              const stdair::EventType::EN_EventType&,
                              const stdair::Count_T&);
    static void addStatus (EventQueue&,
                           const stdair::EventType::EN_EventType&,
                           const stdair::Count_T&);

    static bool hasProgressStatus (const EventQueue&,
                                   const stdair::EventType::EN_EventType&);

    static bool isQueueDone (const EventQueue&);


    static const stdair::Count_T& getQueueSize(const EventQueue&);

    static const stdair::Count_T&
    getExpectedTotalNumberOfEventsToBeGenerated(const EventQueue&);

    static const stdair::Count_T&
    getExpectedTotalNumberOfEventsToBeGenerated(const EventQueue&,
                                                const stdair::EventType::EN_EventType&);

    static const stdair::Count_T&
    getActualTotalNumberOfEventsToBeGenerated(const EventQueue&);

    static const stdair::Count_T&
    getActualTotalNumberOfEventsToBeGenerated(const EventQueue&,
                                              const stdair::EventType::EN_EventType&);

    static const stdair::ProgressStatus& getStatus (const EventQueue&,
                                                    const stdair::EventType::EN_EventType&);

    static const stdair::ProgressStatus& getStatus (const EventQueue&);

  };

}
#endif // __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP
```

25.43 sevmgr/config/sevmgr-paths.hpp.in File Reference

Defines

- #define PACKAGE "@PACKAGE@"
- #define PACKAGE_NAME "@PACKAGE_NAME@"
- #define PACKAGE_VERSION "@PACKAGE_VERSION@"
- #define PREFIXDIR "@prefix@"
- #define EXEC_PREFIX "@exec_prefix@"
- #define BINDIR "@bindir@"
- #define LIBDIR "@libdir@"
- #define LIBEXECDIR "@libexecdir@"
- #define SBINDIR "@sbindir@"
- #define SYSCONFDIR "@sysconfdir@"
- #define INCLUDEDIR "@includedir@"
- #define DATAROOTDIR "@datarootdir@"
- #define DATADIR "@datadir@"
- #define DOCDIR "@docdir@"
- #define MANDIR "@mandir@"
- #define INFODIR "@infodir@"
- #define HTMLDIR "@htmldir@"
- #define PDFDIR "@pdfdir@"
- #define STDAIR_SAMPLE_DIR "@sampledir@"

25.43.1 Define Documentation

25.43.1.1 #define PACKAGE "@PACKAGE@"

Definition at line 4 of file sevmgr-paths.hpp.in.

25.43.1.2 #define PACKAGE_NAME "@PACKAGE_NAME@"

Definition at line 5 of file sevmgr-paths.hpp.in.

Referenced by readConfiguration().

25.43.1.3 #define PACKAGE_VERSION "@PACKAGE_VERSION@"

Definition at line 6 of file sevmgr-paths.hpp.in.

Referenced by readConfiguration().

25.43.1.4 #define PREFIXDIR "@prefix@"

Definition at line 7 of file sevmgr-paths.hpp.in.

Referenced by readConfiguration().

25.43.1.5 #define EXEC_PREFIX "@exec_prefix@"

Definition at line 8 of file sevmgr-paths.hpp.in.

25.43.1.6 #define BINDIR "@bindir@"

Definition at line 9 of file sevmgr-paths.hpp.in.

25.43.1.7 #define LIBDIR "@libdir@"

Definition at line 10 of file sevmgr-paths.hpp.in.

25.43.1.8 #define LIBEXECDIR "@libexecdir@"

Definition at line 11 of file sevmgr-paths.hpp.in.

25.43.1.9 #define SBINDIR "@sbindir@"

Definition at line 12 of file sevmgr-paths.hpp.in.

25.43.1.10 #define SYSCONFDIR "@sysconfdir@"

Definition at line 13 of file sevmgr-paths.hpp.in.

25.43.1.11 #define INCLUDEDIR "@includedir@"

Definition at line 14 of file sevmgr-paths.hpp.in.

25.43.1.12 #define DATAROOTDIR "@datarootdir@"

Definition at line 15 of file sevmgr-paths.hpp.in.

25.43.1.13 #define DATADIR "@datadir@"

Definition at line 16 of file sevmgr-paths.hpp.in.

25.43.1.14 #define DOCDIR "@docdir@"

Definition at line 17 of file sevmgr-paths.hpp.in.

25.43.1.15 #define MANDIR "@mandir@"

Definition at line 18 of file sevmgr-paths.hpp.in.

25.43.1.16 #define INFODIR "@infodir@"

Definition at line 19 of file sevmgr-paths.hpp.in.

25.43.1.17 #define HTMLDIR "@htmldir@"

Definition at line 20 of file sevmgr-paths.hpp.in.

25.43.1.18 #define PDFDIR "@pdfdir@"

Definition at line 21 of file sevmgr-paths.hpp.in.

25.43.1.19 #define STDAIR_SAMPLE_DIR "@sampledir@"

Definition at line 22 of file sevmgr-paths.hpp.in.

25.44 sevmgr-paths.hpp.in

```
#ifndef __SEVMGR_PATHS_HPP__
#define __SEVMGR_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 // __SEVMGR_PATHS_HPP__
```

25.45 sevmgr/factory/FacSEVMGRServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>
```

Namespaces

- namespace SEVMGR

25.46 FacSEVMGRServiceContext.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/service/FacSupervisor.hpp>
// Sevmgr
#include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>

namespace SEVMGR {

  FacSEVMGRServiceContext* FacSEVMGRServiceContext::_instance = NULL;

  // //////////////////////////////////////////////////////////////////////
  FacSEVMGRServiceContext::~FacSEVMGRServiceContext () {
    _instance = NULL;
  }

  // //////////////////////////////////////////////////////////////////////
  FacSEVMGRServiceContext& FacSEVMGRServiceContext::instance () {

    if (_instance == NULL) {
      _instance = new FacSEVMGRServiceContext();
      assert (_instance != NULL);

      stdair::FacSupervisor::instance().registerServiceFactory (_instance);
    }
    return *_instance;
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_ServiceContext& FacSEVMGRServiceContext::create () {
    SEVMGR_ServiceContext* aServiceContext_ptr = NULL;

    aServiceContext_ptr = new SEVMGR_ServiceContext ();
    assert (aServiceContext_ptr != NULL);

    // The new object is added to the Bom pool
    _pool.push_back (aServiceContext_ptr);

    return *aServiceContext_ptr;
  }

}
```

25.47 sevmgr/factory/FacSEVMGRServiceContext.hpp File Reference

```
#include <stdair/service/FacServiceAbstract.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class SEVMGR::FacSEVMGRServiceContext

Namespaces

- namespace SEVMGR

25.48 FacSEVMGRServiceContext.hpp

```
#ifndef __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP
#define __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/service/FacServiceAbstract.hpp>
// Sevmgr
#include <sevmgr/SEVMGR_Types.hpp>

namespace SEVMGR {

  class SEVMGR_ServiceContext;

  class FacSEVMGRServiceContext : public stdair::FacServiceAbstract {
  public:

    static FacSEVMGRServiceContext& instance();

    ~FacSEVMGRServiceContext();

    SEVMGR_ServiceContext& create ();


  protected:
    FacSEVMGRServiceContext () {}

  private:
    static FacSEVMGRServiceContext* _instance;
  };

}
#endif // __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP
```

25.49 sevmgr/python/pysevmgr.cpp File Reference

```
#include <cassert>
#include <stdexcept>
#include <fstream>
#include <sstream>
#include <string>
#include <list>
#include <vector>
#include <boost/python.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
```

Classes

- struct SEVMGR::PYEventQueueManager

Namespaces

- namespace SEVMGR

Functions

- BOOST_PYTHON_MODULE (libpysevmgr)

25.49.1 Function Documentation

25.49.1.1 BOOST_PYTHON_MODULE (libpysevmgr)

Definition at line 152 of file pysevmgr.cpp.

References SEVMGR::PYEventQueueManager::init(), and SEVMGR::PYEventQueueManager::sevmgr().

25.50 pysevmgr.cpp

```
// STL
#include <cassert>
#include <stdexcept>
#include <fstream>
#include <sstream>
#include <string>
#include <list>
#include <vector>
// Boost String
#include <boost/python.hpp>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Service.hpp>

namespace SEVMGR {

  struct PYEventQueueManager {
  public:
    std::string sevmgr() {
      std::ostringstream oStream;

      // Sanity check
      if (_logOutputStream == NULL) {
        oStream << "The log filepath is not valid." << std::endl;
        return oStream.str();
      }
      assert (_logOutputStream != NULL);

      try {

        // DEBUG
        *_logOutputStream << "Default service" << std::endl;

        if (_sevmgrService == NULL) {
          oStream << "The Sevmgr service has not been initialised, "
                  << "i.e., the init() method has not been called "
                  << "correctly on the PYEventQueueManager object. Please "
                  << "check that all the parameters are not empty and "
                  << "point to actual files.";
          *_logOutputStream << oStream.str();
          return oStream.str();
        }
        assert (_sevmgrService != NULL);

        // Do the sevmgr
        _sevmgrService->buildSampleBom();

        // DEBUG
        *_logOutputStream << "Default service returned" << std::endl;

        // DEBUG
        *_logOutputStream << "Sevmgr output: " << oStream.str() << std::endl;

      } catch (const stdair::RootException& eSevmgrError) {
        *_logOutputStream << "Sevmgr error: "  << eSevmgrError.what()
                          << std::endl;

      } catch (const std::exception& eStdError) {
        *_logOutputStream << "Error: "  << eStdError.what() << std::endl;

      } catch (...) {
        *_logOutputStream << "Unknown error" << std::endl;
      }

      return oStream.str();
    }

  public:
    PYEventQueueManager() : _sevmgrService (NULL), _logOutputStream (NULL) {
    }

    PYEventQueueManager (const PYEventQueueManager& iPYEventQueueManager)
      : _sevmgrService (iPYEventQueueManager._sevmgrService),
        _logOutputStream (iPYEventQueueManager._logOutputStream) {
    }

    ~PYEventQueueManager() {
      _sevmgrService = NULL;
      _logOutputStream = NULL;
    }

    bool init (const std::string& iLogFilepath,
               const std::string& iDBUser, const std::string& iDBPasswd,
               const std::string& iDBHost, const std::string& iDBPort,
               const std::string& iDBDBName) {
      bool isEverythingOK = true;

      try {

        // Check that the file path given as input corresponds to an actual file
        const bool isWriteable = (iLogFilepath.empty() == false);
        // stdair::BasFileMgr::isWriteable (iLogFilepath);
        if (isWriteable == false) {
          isEverythingOK = false;
          return isEverythingOK;
        }

        // Set the log parameters
        _logOutputStream = new std::ofstream;
        assert (_logOutputStream != NULL);

        // Open and clean the log outputfile
        _logOutputStream->open (iLogFilepath.c_str());
        _logOutputStream->clear();

        // DEBUG
        *_logOutputStream << "Python wrapper initialisation" << std::endl;
        const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
                                               *_logOutputStream);

        // Initialise the context
        stdair::BasDBParams lDBParams (iDBUser, iDBPasswd, iDBHost, iDBPort,
                                       iDBDBName);
        _sevmgrService = new SEVMGR_Service (lLogParams, lDBParams);

        // DEBUG
        *_logOutputStream << "Python wrapper initialised" << std::endl;

      } catch (const stdair::RootException& eSevmgrError) {
        *_logOutputStream << "Sevmgr error: "  << eSevmgrError.what()
                          << std::endl;

      } catch (const std::exception& eStdError) {
        *_logOutputStream << "Error: "  << eStdError.what() << std::endl;

      } catch (...) {
        *_logOutputStream << "Unknown error" << std::endl;
      }

      return isEverythingOK;
    }

  private:
    SEVMGR_Service* _sevmgrService;
    std::ofstream* _logOutputStream;
  };

}

// //////////////////////////////////////////////////////////////
BOOST_PYTHON_MODULE(libpysevmgr) {
  boost::python::class_<SEVMGR::PYEventQueueManager> ("PYEventQueueManager")
    .def ("sevmgr", &SEVMGR::PYEventQueueManager::sevmgr)
    .def ("init", &SEVMGR::PYEventQueueManager::init);
}
```

25.51 sevmgr/service/SEVMGR_Service.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
#include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
#include <sevmgr/command/EventQueueManager.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/bom/BomJSONExport.hpp>
```

Namespaces

- namespace SEVMGR

25.52 SEVMGR_Service.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#include <boost/make_shared.hpp>
// StdAir
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
// Sevmgr
#include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
#include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
#include <sevmgr/command/EventQueueManager.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/bom/BomJSONExport.hpp>

namespace SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_Service::SEVMGR_Service() : _sevmgrServiceContext (NULL) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_Service::SEVMGR_Service (const SEVMGR_Service& iService)
    : _sevmgrServiceContext (NULL) {
    assert (false);
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams& iLogParams,
                                  const stdair::BasDBParams& iDBParams)
    : _sevmgrServiceContext (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 SEvMgr service context
    // \note SEvMgr owns the STDAIR service resources here.
    const bool ownStdairService = true;
    addStdAirService (lSTDAIR_Service_ptr, ownStdairService);

    // Initialise the (remaining of the) context
    initSevmgrService();
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams& iLogParams)
    : _sevmgrServiceContext (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 SEvMgr service context
    // \note SEvMgr owns the STDAIR service resources here.
    const bool ownStdairService = true;
    addStdAirService (lSTDAIR_Service_ptr, ownStdairService);

    // Initialise the (remaining of the) context
    initSevmgrService();
  }

  // ////////////////////////////////////////////////////////////////////
  SEVMGR_Service::
  SEVMGR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
    : _sevmgrServiceContext (NULL) {

    // Initialise the service context
    initServiceContext();

    // Add the StdAir service context to the SEvMgr service context
    // \note SEvMgr does not own the STDAIR service resources here.
    const bool doesNotOwnStdairService = false;
    addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);

    // Initialise the context
    initSevmgrService();
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_Service::~SEVMGR_Service() {
    // Delete/Clean all the objects from memory
    finalise();
  }

  // ////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::finalise() {
    assert (_sevmgrServiceContext != NULL);
    // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
    _sevmgrServiceContext->reset();
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::initServiceContext() {
    // Initialise the service context
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
      FacSEVMGRServiceContext::instance().create();
    _sevmgrServiceContext = &lSEVMGR_ServiceContext;
  }

  // ////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::
  addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
                    const bool iOwnStdairService) {
    // Retrieve the SEvMgr service context
    assert (_sevmgrServiceContext != NULL);
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
      *_sevmgrServiceContext;

    // Store the STDAIR service object within the (SEvMgr) service context
    lSEVMGR_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
                                              iOwnStdairService);
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::STDAIR_ServicePtr_T SEVMGR_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);
    assert (lSTDAIR_Service_ptr != NULL);

    return lSTDAIR_Service_ptr;
  }

  // //////////////////////////////////////////////////////////////////////
  stdair::STDAIR_ServicePtr_T SEVMGR_Service::
  initStdAirService (const stdair::BasLogParams& iLogParams) {

    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      boost::make_shared<stdair::STDAIR_Service> (iLogParams);
    assert (lSTDAIR_Service_ptr != NULL);

    return lSTDAIR_Service_ptr;
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::initSevmgrService() {
    // Do nothing at this stage. A sample BOM tree may be built by
    // calling the buildSampleBom() method
  }

  // /////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::buildSampleQueue() {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
                                                    "not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the StdAir service context
    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();

    // Retrieve the EventQueue
    EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the building process to the dedicated command
    EventQueueManager::buildSampleQueue (lSTDAIR_Service_ptr, lEventQueue);

  }

  // //////////////////////////////////////////////////////////////////////
  stdair::BookingRequestStruct SEVMGR_Service::
  buildSampleBookingRequest(const bool isForCRS) {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
                                                    "not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the StdAir service context
    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();

    // Delegate the booking request building to the dedicated service
    stdair::BookingRequestStruct oBookingRequest =
      EventQueueManager::buildSampleBookingRequest (lSTDAIR_Service_ptr,
                                                    isForCRS);

    return oBookingRequest;
  }

  // //////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::describeKey() const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
                                                    "not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue
    EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the key display to the dedicated command
    return EventQueueManager::describeKey(lEventQueue);
  }

  // //////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::list () const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
                                                    "not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue
    EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue ();

    // Delegate the key display to the dedicated command
    return EventQueueManager::list (lEventQueue);
  }

  // //////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::
  list (const stdair::EventType::EN_EventType& iEventType) const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
                                                    "not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue
    EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue ();

    // Delegate the key display to the dedicated command
    return EventQueueManager::list (lEventQueue, iEventType);
  }

  // //////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::
  jsonHandler (const stdair::JSONString& iJSONString) const {

    //
    // Extract from the JSON-ified string the command
    //
    stdair::JSonCommand::EN_JSonCommand lEN_JSonCommand;
    const bool hasCommandBeenRetrieved =
      stdair::BomJSONImport::jsonImportCommand (iJSONString,
                                                lEN_JSonCommand);

    if (hasCommandBeenRetrieved == false) {
      // Return an error JSON-ified string
      std::ostringstream oErrorStream;
      oErrorStream << "{\"error\": \"Wrong JSON-ified string: "
                   << "the command is not understood.\"}";
      return oErrorStream.str();
    }
    assert (hasCommandBeenRetrieved == true);

    //
    // Dispatch the command to the right JSon service handler
    //
    switch (lEN_JSonCommand) {
    case stdair::JSonCommand::EVENT_LIST:{

      //
      // Try to extract the event type from the JSON-ified string
      //
      stdair::EventType::EN_EventType lEN_EventType;
      const bool hasEventTypeBeenRetrieved =
        stdair::BomJSONImport::jsonImportEventType (iJSONString,
                                                    lEN_EventType);

      if (hasEventTypeBeenRetrieved == true) {
        return jsonExportEventQueue (lEN_EventType);
      }
      return jsonExportEventQueue ();
    }
    default: {
        // Return an Error string
        std::ostringstream lErrorCmdMessage;
        const std::string& lCommandStr =
          stdair::JSonCommand::getLabel(lEN_JSonCommand);
        lErrorCmdMessage << "{\"error\": \"The command '" << lCommandStr
                         << "' is not handled by the DSim service.\"}";
        return lErrorCmdMessage.str();
        break;
      }
    }

    // Return an error JSON-ified string
    assert (false);
    std::string lJSONDump ("{\"error\": \"Wrong JSON-ified string\"}");
    return lJSONDump;

  }

  // //////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::
  jsonExportEventQueue (const stdair::EventType::EN_EventType& iEventType) const
{

    std::ostringstream oStr;

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the StdAir service context
    stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
      lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();

    // Retrieve the event queue
    const EventQueue& lEventQueue =
      lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the JSON export to the dedicated command
    BomJSONExport::jsonExportEventQueue (lSTDAIR_Service_ptr, oStr,
                                         lEventQueue, iEventType);
    return oStr.str();

  }

  // ////////////////////////////////////////////////////////////////////
  std::string SEVMGR_Service::
  jsonExportEvent (const stdair::EventStruct& iEvent) const {

    std::ostringstream oStr;

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);

    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the STDAIR service object from the (SEvMgr) service context
    stdair::STDAIR_Service& lSTDAIR_Service =
      lSEVMGR_ServiceContext.getSTDAIR_Service();

    // Delegate the JSON export to the dedicated service
    oStr << lSTDAIR_Service.jsonExportEventObject (iEvent);

    return oStr.str();

  }

  // ////////////////////////////////////////////////////////////////////
  stdair::ProgressStatusSet SEVMGR_Service::
  popEvent (stdair::EventStruct& iEventStruct) const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue object instance
    EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the call to the dedicated command
    return EventQueueManager::popEvent(lQueue, iEventStruct);
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::
  run (stdair::EventStruct& iEventStruct) const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue object instance
    EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the call to the dedicated command
    EventQueueManager::run (lQueue, iEventStruct);

  }

  // //////////////////////////////////////////////////////////////////////
  bool SEVMGR_Service::
  select (stdair::EventStruct& iEventStruct,
          const stdair::DateTime_T& iEventDateTime) const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue object instance
    EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the call to the dedicated command
    return EventQueueManager::select (lQueue, iEventStruct, iEventDateTime);

  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_Service::
  updateStatus (const stdair::EventType::EN_EventType& iEventType,
                const stdair::Count_T& iEventCount) const {

    // Retrieve the SEvMgr service context
    if (_sevmgrServiceContext == NULL) {
      throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                    "has not been initialised");
    }
    assert (_sevmgrServiceContext != NULL);
    SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

    // Retrieve the event queue object instance
    EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

    // Delegate the call to the dedicated command
    EventQueueManager::updateStatus (lQueue, iEventType, iEventCount);
  }

  // //////////////////////////////////////////////////////////////////////
```

```
void SEVMGR_Service::
addStatus (const stdair::EventType::EN_EventType& iEventType,
           const stdair::Count_T& iEventCount) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  EventQueueManager::addStatus (lQueue, iEventType, iEventCount);
}

// //////////////////////////////////////////////////////////////////////
bool SEVMGR_Service::isQueueDone() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Calculates whether the event queue has been fully emptied
  const bool isQueueDone = EventQueueManager::isQueueDone(lQueue);

  //
  return isQueueDone;
}

// //////////////////////////////////////////////////////////////////////
bool SEVMGR_Service::hasProgressStatus(const stdair::EventType::EN_EventType& i
EventType) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Calculates whether the event queue has been fully emptied
  const bool hasProgressStatus =
     EventQueueManager::hasProgressStatus (lQueue, iEventType);

  //
  return hasProgressStatus;
}

// //////////////////////////////////////////////////////////////////////
const stdair::Count_T& SEVMGR_Service::getQueueSize() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated command
  return EventQueueManager::getQueueSize(lQueue);
}

// //////////////////////////////////////////////////////////////////////
void SEVMGR_Service::reset() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated command
  EventQueueManager::reset (lQueue);
}

// ///////////////////////////////////////////////////////////////////////
EventQueue& SEVMGR_Service::getEventQueue() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);

  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  return lSEVMGR_ServiceContext.getEventQueue();
}

// //////////////////////////////////////////////////////////////////////
void SEVMGR_Service::addEvent(stdair::EventStruct& iEventStruct) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated command
  EventQueueManager::addEvent (lQueue, iEventStruct);
}

// //////////////////////////////////////////////////////////////////////
const stdair::Count_T& SEVMGR_Service::
getExpectedTotalNumberOfEventsToBeGenerated() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getExpectedTotalNumberOfEventsToBeGenerated (lQueue
);
}

// //////////////////////////////////////////////////////////////////////
const stdair::Count_T& SEVMGR_Service::
getExpectedTotalNumberOfEventsToBeGenerated(const stdair::EventType::EN_EventTy
pe& iEventType) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getExpectedTotalNumberOfEventsToBeGenerated (lQueue
,
                                                                         iEvent
Type);
}

// //////////////////////////////////////////////////////////////////////
const stdair::Count_T& SEVMGR_Service::
getActualTotalNumberOfEventsToBeGenerated() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getActualTotalNumberOfEventsToBeGenerated(lQueue);

}

// //////////////////////////////////////////////////////////////////////
const stdair::Count_T& SEVMGR_Service::
getActualTotalNumberOfEventsToBeGenerated(const stdair::EventType::EN_EventType
& iEventType) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getActualTotalNumberOfEventsToBeGenerated(lQueue,
                                                                       iEventTyp
e);

}

const stdair::STDAIR_Service& SEVMGR_Service::getSTDAIR_Service() const {

    // Retrieve the StdAir service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  const stdair::STDAIR_Service& lSTDAIR_Service =
    _sevmgrServiceContext->getSTDAIR_Service();

  //
  return lSTDAIR_Service;
}

const stdair::ProgressStatus& SEVMGR_Service::getStatus() const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getStatus(lQueue);

}

const stdair::ProgressStatus& SEVMGR_Service::
getStatus(const stdair::EventType::EN_EventType& iEventType) const {

  // Retrieve the SEvMgr service context
  if (_sevmgrServiceContext == NULL) {
    throw stdair::NonInitialisedServiceException ("The SEvMgr service "
                                                  "has not been initialised");
  }
  assert (_sevmgrServiceContext != NULL);
  SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;

  // Retrieve the event queue object instance
  EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();

  // Delegate the call to the dedicated function
  return EventQueueManager::getStatus(lQueue, iEventType);

}


}
```

25.53 sevmgr/service/SEVMGR_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/factory/FacBom.hpp>
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>
```

Namespaces

- namespace SEVMGR

25.54 SEVMGR_ServiceContext.cpp

```
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/STDAIR_Service.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/factory/FacBom.hpp>
// SEvMgr
#include <sevmgr/basic/BasConst_EventQueueManager.hpp>
#include <sevmgr/bom/EventQueue.hpp>
#include <sevmgr/service/SEVMGR_ServiceContext.hpp>

namespace SEVMGR {

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_ServiceContext::SEVMGR_ServiceContext()
    : _eventQueue (NULL) {
    init();
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_ServiceContext::
  SEVMGR_ServiceContext (const SEVMGR_ServiceContext& iServiceContext)
    : _eventQueue (iServiceContext._eventQueue) {
  }

  // //////////////////////////////////////////////////////////////////////
  SEVMGR_ServiceContext::~SEVMGR_ServiceContext() {
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_ServiceContext::init() {
    //
    initEventQueue();
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_ServiceContext::initEventQueue() {

    // The event queue key is just a string. For now, it is not used.
    const EventQueueKey lKey ("EQ01");

    // Create an EventQueue object instance
    EventQueue& lEventQueue = stdair::FacBom<EventQueue>::instance().create (lKey
);

    // Store the event queue object
    _eventQueue = &lEventQueue;
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string SEVMGR_ServiceContext::shortDisplay() const {
    std::ostringstream oStr;
    oStr << "SEVMGR_ServiceContext -- Owns StdAir service: "
         << _ownStdairService;
    if (_eventQueue != NULL) {
      oStr << " -- Queue: " << _eventQueue->toString();
    }
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string SEVMGR_ServiceContext::display() const {
    std::ostringstream oStr;
    oStr << shortDisplay();
    return oStr.str();
  }

  // //////////////////////////////////////////////////////////////////////
  const std::string SEVMGR_ServiceContext::describe() const {
    return shortDisplay();
  }

  // //////////////////////////////////////////////////////////////////////
  void SEVMGR_ServiceContext::reset() {

    // The shared_ptr<>::reset() method drops the refcount by one.
    // If the count result is dropping to zero, the resource pointed to
    // by the shared_ptr<> will be freed.

    // Reset the stdair shared pointer
    _stdairService.reset();
  }

  // //////////////////////////////////////////////////////////////////////
  EventQueue& SEVMGR_ServiceContext::getEventQueue() const {
    assert (_eventQueue != NULL);
    return *_eventQueue;
  }

}
```

25.55 sevmgr/service/SEVMGR_ServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class SEVMGR::SEVMGR_ServiceContext

  Class holding the context of the Sevmgr services.

Namespaces

- namespace stdair

  Forward declarations.

- namespace SEVMGR

25.56 SEVMGR_ServiceContext.hpp

```
#ifndef __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP
#define __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Types.hpp>

namespace stdair {
  class FacBomManager;
  template <typename BOM> class FacBom;
}

namespace SEVMGR {

  class EventQueue;

  class SEVMGR_ServiceContext : public stdair::ServiceAbstract {
    friend class SEVMGR_Service;
    friend class FacSEVMGRServiceContext;

  private:
    // ///////// Getters //////////
    stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
      return _stdairService;
    }

    stdair::STDAIR_Service& getSTDAIR_Service() const {
      assert (_stdairService != NULL);
      return *_stdairService;
    }

    const bool getOwnStdairServiceFlag() const {
      return _ownStdairService;
    }

    EventQueue& getEventQueue() const;


  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:

    SEVMGR_ServiceContext();
    SEVMGR_ServiceContext (const SEVMGR_ServiceContext&);

    ~SEVMGR_ServiceContext();

    void reset();

    void init();

    void initEventQueue();


  private:
    // /////////////// Children ///////////////
    stdair::STDAIR_ServicePtr_T _stdairService;

    bool _ownStdairService;


  private:
    // ////////////// Attributes ////////////////
    EventQueue* _eventQueue;
  };

}
#endif // __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP
```

25.57 sevmgr/SEVMGR_Exceptions.hpp File Reference

```
#include <exception>
#include <stdair/stdair_exceptions.hpp>
```

Classes

- class SEVMGR::SEvMgrException
- class SEVMGR::EventQueueException

Namespaces

- namespace SEVMGR

25.58 SEVMGR_Exceptions.hpp

```
#ifndef __SEVMGR_SEVMGR_EXCEPTIONS_HPP
#define __SEVMGR_SEVMGR_EXCEPTIONS_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <exception>
// StdAir
#include <stdair/stdair_exceptions.hpp>

namespace SEVMGR {

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

  class EventQueueException : public SEvMgrException {
  public:
    EventQueueException (const std::string& iWhat) : SEvMgrException (iWhat) {}
  };


}
#endif // __SEVMGR_SEVMGR_EXCEPTIONS_HPP

```

25.59 sevmgr/SEVMGR_Service.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/EventStruct.hpp>
```

Classes

- class SEVMGR::SEVMGR_Service

  class holding the services related to Travel Demand Generation.

Namespaces

- namespace stdair

  Forward declarations.

- namespace SEVMGR

25.60 SEVMGR_Service.hpp

```
#ifndef __SEVMGR_SEVMGR_SERVICE_HPP
#define __SEVMGR_SEVMGR_SERVICE_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/EventStruct.hpp>

// Forward declarations
namespace stdair {
  struct ProgressStatusSet;
  struct BasLogParams;
  struct BasDBParams;
  struct BookingRequestStruct;
}

namespace SEVMGR {

  class SEVMGR_ServiceContext;
  class EventQueue;
  //struct EventStruct;

  class SEVMGR_Service {
  public:
    // ////////////////// Constructors and Destructors //////////////////
    SEVMGR_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);

    SEVMGR_Service (const stdair::BasLogParams&);

    SEVMGR_Service (stdair::STDAIR_ServicePtr_T);

    ~SEVMGR_Service();


  public:
    // ////////////////// Business support methods //////////////////
    void buildSampleQueue();

    stdair::BookingRequestStruct buildSampleBookingRequest(const bool isForCRS = false);

    stdair::ProgressStatusSet popEvent (stdair::EventStruct&) const;


    void run (stdair::EventStruct&) const;

    bool select (stdair::EventStruct&,
                 const stdair::DateTime_T&) const;

    template<class EventGenerator>
    void addEventGenerator(EventGenerator& iEventGenerator) const;

    void addEvent(stdair::EventStruct&) const;

    void reset() const;

    void updateStatus (const stdair::EventType::EN_EventType&,
                       const stdair::Count_T&) const;

     void addStatus (const stdair::EventType::EN_EventType&,
                       const stdair::Count_T&) const;

    bool isQueueDone() const;

    bool hasProgressStatus(const stdair::EventType::EN_EventType&) const;

    /* @brief Get a reference on the EventQueue object.
     *
     * @return EventQueue& Reference on the EventQueue.
     */
    EventQueue& getEventQueue() const;

    const stdair::Count_T& getQueueSize() const;

    template<class EventGenerator, class Key>
    EventGenerator& getEventGenerator(const Key& iKey) const;

    template<class EventGenerator, class Key>
    bool hasEventGenerator(const Key& iKey) const;

    template<class EventGenerator>
    const std::list<EventGenerator*> getEventGeneratorList() const;

    template<class EventGenerator>
    bool hasEventGeneratorList() const;

    const stdair::Count_T& getExpectedTotalNumberOfEventsToBeGenerated() const;

    const stdair::Count_T&
    getExpectedTotalNumberOfEventsToBeGenerated(const stdair::EventType::EN_Event
Type&) const;

    const stdair::Count_T& getActualTotalNumberOfEventsToBeGenerated() const;

    const stdair::Count_T&
    getActualTotalNumberOfEventsToBeGenerated(const stdair::EventType::EN_EventTy
pe&) const;

    const stdair::ProgressStatus& getStatus () const;

    const stdair::ProgressStatus& getStatus (const stdair::EventType::EN_EventTyp
e&) const;

  public:
    // //////////////// Display support methods /////////////////

    std::string describeKey() const;

    std::string list () const;

    std::string list (const stdair::EventType::EN_EventType&) const;

  public:
    // //////////////// Export support methods /////////////////
    std::string jsonHandler (const stdair::JSONString&) const;

    std::string jsonExportEventQueue (const stdair::EventType::EN_EventType& =
                                      stdair::EventType::LAST_VALUE) const;

    std::string jsonExportEvent (const stdair::EventStruct&) const;

  private:
    // ////////////////// Constructors and Destructors /////////////////
    SEVMGR_Service();

    SEVMGR_Service (const SEVMGR_Service&);

    void initServiceContext();

    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 ioSTDAIR_ServicePtr,
                           const bool iOwnStdairService);

    void initSevmgrService();

    void finalise();

  private:
    // ////////////////// Getters /////////////////
    const stdair::STDAIR_Service& getSTDAIR_Service() const;

  private:
    // ///////// Service Context /////////
    SEVMGR_ServiceContext* _sevmgrServiceContext;
  };

}
#endif // __SEVMGR_SEVMGR_SERVICE_HPP
```

25.61 sevmgr/SEVMGR_Types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <sevmgr/SEVMGR_Exceptions.hpp>
```

Namespaces

- namespace SEVMGR

Typedefs

- typedef boost::shared_ptr< SEVMGR_Service > SEVMGR::SEVMGR_ServicePtr_T
- typedef std::string SEVMGR::EventQueueID_T
- typedef std::map< stdair::EventType::EN_EventType, stdair::ProgressStatus > SEVMGR::ProgressStatusMap_T

25.62 SEVMGR_Types.hpp

```
#ifndef __SEVMGR_SEVMGR_TYPES_HPP
#define __SEVMGR_SEVMGR_TYPES_HPP

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// Boost
#include <boost/shared_ptr.hpp>
// Stdair
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
// Sevmgr
#include <sevmgr/SEVMGR_Exceptions.hpp>

namespace SEVMGR {

  // Forward declarations
  class SEVMGR_Service;

  // //////// Type definitions specific to to Sevmgr /////////
  typedef boost::shared_ptr<SEVMGR_Service> SEVMGR_ServicePtr_T;

  typedef std::string EventQueueID_T;

  typedef std::map<stdair::EventType::EN_EventType,
                   stdair::ProgressStatus> ProgressStatusMap_T;

}
#endif // __SEVMGR_SEVMGR_TYPES_HPP

```

25.63 sevmgr/ui/cmdline/sevmgr.cpp File Reference

25.64 sevmgr.cpp

```

// STL
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
// Boost (Extended STL)
#include <boost/program_options.hpp>
#include <boost/tokenizer.hpp>
#include <boost/regex.hpp>
#include <boost/swap.hpp>
#include <boost/algorithm/string/case_conv.hpp>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/bom/EventStruct.hpp>
// GNU Readline Wrapper
#include <stdair/ui/cmdline/SReadline.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/config/sevmgr-paths.hpp>

// //////// Constants //////
const std::string K_SEVMGR_DEFAULT_LOG_FILENAME ("sevmgr.log");

const int K_SEVMGR_EARLY_RETURN_STATUS = 99;

typedef std::vector<std::string> TokenList_T;

struct Command_T {
  typedef enum {
    NOP = 0,
    QUIT,
    HELP,
    LIST,
    DISPLAY,
    SELECT,
    NEXT,
    RUN,
    JSON_LIST,
    JSON_DISPLAY,
    LAST_VALUE
  } Type_T;
};

// ///////// 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;
}

int readConfiguration (int argc, char* argv[], std::string& ioLogFilename) {
  // 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()
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(
K_SEVMGR_DEFAULT_LOG_FILENAME),
     "Filename for the logs")
    ;

  // 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 ("sevmgr.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_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("version")) {
    std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
    return K_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("prefix")) {
    std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
    return K_SEVMGR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}

// //////////////////////////////////////////////////////////////////////
void initReadline (swift::SReadline& ioInputReader) {

  // Prepare the list of my own completers
  std::vector<std::string> Completers;

  // The following is supported:
  // - "identifiers"
  // - special identifier %file - means to perform a file name completion
  Completers.push_back ("help");
  Completers.push_back ("list");
  Completers.push_back ("list BookingRequest");
  Completers.push_back ("list BreakPoint");
  Completers.push_back ("select %date %time");
  Completers.push_back ("display");
  Completers.push_back ("next");
  Completers.push_back ("run");
  Completers.push_back ("json_list");
  Completers.push_back ("json_display");
  Completers.push_back ("quit");


  // Now register the completers.
  // Actually it is possible to re-register another set at any time
  ioInputReader.RegisterCompletions (Completers);
}

// //////////////////////////////////////////////////////////////////////
void parseEventDateTime (const TokenList_T& iTokenList,
                         stdair::Date_T& ioEventDate,
                         stdair::Duration_T& ioEventTime) {
  //
  const std::string kMonthStr[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                     "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
  //
  unsigned short ioEventDateYear = ioEventDate.year();
  unsigned short ioEventDateMonth = ioEventDate.month();
  std::string ioEventDateMonthStr = kMonthStr[ioEventDateMonth-1];
  unsigned short ioEventDateDay = ioEventDate.day();
  //
  unsigned short ioEventTimeHours = ioEventTime.hours();
  unsigned short ioEventTimeMinutes = ioEventTime.minutes();
  unsigned short ioEventTimeSeconds = ioEventTime.seconds();

  // Interpret the user input
  if (iTokenList.empty() == false) {

    // Read the date year
    TokenList_T::const_iterator itTok = iTokenList.begin();

    // Read the year for the event date
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          ioEventDateYear = boost::lexical_cast<unsigned short> (*itTok);
          if (ioEventDateYear < 100) {
            ioEventDateYear += 2000;
          }

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The year of the event date ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventDateYear << ") is kept. " << std::endl;
          return;
        }
      }

    } else {
      return;
    }

    // Read the month for the event date
    ++itTok;
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          const boost::regex lMonthRegex ("^(\\d{1,2})$");
          const bool isMonthANumber = regex_match (*itTok, lMonthRegex);

          if (isMonthANumber == true) {
            const unsigned short lMonth =
              boost::lexical_cast<unsigned short> (*itTok);
            if (lMonth > 12) {
              throw boost::bad_lexical_cast();
            }
            ioEventDateMonthStr = kMonthStr[lMonth-1];

        } else {
            const std::string lMonthStr (*itTok);
            if (lMonthStr.size() < 3) {
              throw boost::bad_lexical_cast();
            }
            std::string lMonthStr1 (lMonthStr.substr (0, 1));
            boost::algorithm::to_upper (lMonthStr1);
            std::string lMonthStr23 (lMonthStr.substr (1, 2));
            boost::algorithm::to_lower (lMonthStr23);
            ioEventDateMonthStr = lMonthStr1 + lMonthStr23;
          }

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The month of the event date ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventDateMonthStr << ") is kept. " << std::endl;
          return;
        }
      }
    }

    // Read the day for the event date
    ++itTok;
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          ioEventDateDay = boost::lexical_cast<unsigned short> (*itTok);

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The day of the event date ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventDateDay << ") is kept. " << std::endl;
          return;
        }
      }

    } else {
      return;
    }

    // Re-compose the event date
    std::ostringstream lEventDateStr;
    lEventDateStr << ioEventDateYear << "-" << ioEventDateMonthStr
```

```
                    << "-" << ioEventDateDay;

    try {

      ioEventDate =
        boost::gregorian::from_simple_string (lEventDateStr.str());

    } catch (boost::gregorian::bad_month& eCast) {
      std::cerr << "The event date ('" << lEventDateStr.str()
                << "') cannot be understood. The default value ("
                << ioEventDate << ") is kept. " << std::endl;
      return;
    }

    // Read the hours of the event time
    ++itTok;
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          ioEventTimeHours = boost::lexical_cast<unsigned short> (*itTok);

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The hours of the event time ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventTimeHours << ") is kept. " << std::endl;
          return;
        }
      }

    } else {
      return;
    }

    // Read the minutes of the event time
    ++itTok;
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          ioEventTimeMinutes = boost::lexical_cast<unsigned short> (*itTok);

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The minutes of the event time ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventTimeMinutes << ") is kept. " << std::endl;
          return;
        }
      }

    } else {
      return;
    }

    // Read the seconds of the event time
    ++itTok;
    if (itTok != iTokenList.end()) {

      if (itTok->empty() == false) {
        try {

          ioEventTimeSeconds = boost::lexical_cast<unsigned short> (*itTok);

        } catch (boost::bad_lexical_cast& eCast) {
          std::cerr << "The seconds of the event time ('" << *itTok
                    << "') cannot be understood. The default value ("
                    << ioEventTimeSeconds << ") is kept. " << std::endl;
          return;
        }
      }

    } else {
      return;
    }

    // Re-compose the event time
    std::ostringstream lEventTimeStr;
    lEventTimeStr << ioEventTimeHours << ":" << ioEventTimeMinutes
                  << ":" << ioEventTimeSeconds;

    try {

      ioEventTime =
        boost::posix_time::duration_from_string (lEventTimeStr.str());

    } catch (boost::gregorian::bad_month& eCast) {
      std::cerr << "The event time ('" << lEventTimeStr.str()
                << "') cannot be understood. The default value ("
                << ioEventTime << ") is kept. " << std::endl;
      return;
    }

  }

}

// //////////////////////////////////////////////////////////////////////
Command_T::Type_T extractCommand (TokenList_T& ioTokenList) {
  Command_T::Type_T oCommandType = Command_T::LAST_VALUE;

  // Interpret the user input
  if (ioTokenList.empty() == false) {
    TokenList_T::iterator itTok = ioTokenList.begin();
    std::string lCommand (*itTok);
    boost::algorithm::to_lower (lCommand);

    if (lCommand == "help") {
      oCommandType = Command_T::HELP;

    } else if (lCommand == "list") {
      oCommandType = Command_T::LIST;

    } else if (lCommand == "display") {
      oCommandType = Command_T::DISPLAY;

    } else if (lCommand == "select") {
      oCommandType = Command_T::SELECT;

    } else if (lCommand == "next") {
      oCommandType = Command_T::NEXT;

    } else if (lCommand == "run") {
      oCommandType = Command_T::RUN;

    } else if (lCommand == "json_list") {
      oCommandType = Command_T::JSON_LIST;

    } else if (lCommand == "json_display") {
      oCommandType = Command_T::JSON_DISPLAY;

    } else if (lCommand == "quit") {
      oCommandType = Command_T::QUIT;
    }

    // Remove the first token (the command), as the corresponding information
    // has been extracted in the form of the returned command type enumeration
    ioTokenList.erase (itTok);

  } else {
    oCommandType = Command_T::NOP;
  }

  return oCommandType;
}

// ///////////////////////////////////////////////////////////
std::string toString (const TokenList_T& iTokenList) {
  std::ostringstream oStr;

  // Re-create the string with all the tokens, trimmed by read-line
  unsigned short idx = 0;
  for (TokenList_T::const_iterator itTok = iTokenList.begin();
       itTok != iTokenList.end(); ++itTok, ++idx) {
    if (idx != 0) {
      oStr << " ";
    }
    oStr << *itTok;
  }

  return oStr.str();
}

// ///////////////////////////////////////////////////////////
TokenList_T extractTokenList (const TokenList_T& iTokenList,
                              const std::string& iRegularExpression) {
  TokenList_T oTokenList;

  // Re-create the string with all the tokens (which had been trimmed
  // by read-line)
  const std::string lFullLine = toString (iTokenList);

  // See the caller for the regular expression
  boost::regex expression (iRegularExpression);

  std::string::const_iterator start = lFullLine.begin();
  std::string::const_iterator end = lFullLine.end();

  boost::match_results<std::string::const_iterator> what;
  boost::match_flag_type flags = boost::match_default | boost::format_sed;
  regex_search (start, end, what, expression, flags);

  // Put the matched strings in the list of tokens to be returned back
  // to the caller
  const unsigned short lMatchSetSize = what.size();
  for (unsigned short matchIdx = 1; matchIdx != lMatchSetSize; ++matchIdx) {
    const std::string lMatchedString (std::string (what[matchIdx].first,
                                                   what[matchIdx].second));
    //if (lMatchedString.empty() == false) {
    oTokenList.push_back (lMatchedString);
    //}
  }

  // DEBUG
  // std::cout << "After (token list): " << oTokenList << std::endl;

  return oTokenList;
}

// ///////////////////////////////////////////////////////////
```

```
TokenList_T extractTokenListForDateTime (const TokenList_T& iTokenList) {
  const std::string lRegEx("^([[:digit:]]{2,4})?[/-]?[[:space:]]*"
                           "([[:alpha:]]{3}|[[:digit:]]{1,2})?[/-]?[[:space:]]*"
                           "([[:digit:]]{1,2})?[[:space:]]*"
                           "([[:digit:]]{1,2})?[:-]?[[:space:]]*"
                           "([[:alpha:]]{3}|[[:digit:]]{1,2})?[:-]?[[:space:]]*"
                           "([[:digit:]]{1,2})?[[:space:]]*$");

  //
  const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegEx);
  return oTokenList;
}

// ///////// M A I N ////////////
int main (int argc, char* argv[]) {

  // Readline history
  const unsigned int lHistorySize (100);
  const std::string lHistoryFilename ("sevmgr.hist");
  const std::string lHistoryBackupFilename ("sevmgr.hist.bak");

  // Default parameters for the interactive session
  stdair::EventStruct lCurrentInteractiveEventStruct;
  stdair::DateTime_T lCurrentInteractiveDateTime;
  stdair::EventType::EN_EventType lCurrentInteractiveEventType;

  // Output log File
  stdair::Filename_T lLogFilename;

  // Call the command-line option parser
  const int lOptionParserStatus = readConfiguration (argc, argv, lLogFilename);

  if (lOptionParserStatus == K_SEVMGR_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 inventory service
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  SEVMGR::SEVMGR_Service sevmgrService (lLogParams);

  // Build the sample event queue.
  sevmgrService.buildSampleQueue();

  // Pop out the first event from the queue.
  sevmgrService.popEvent(lCurrentInteractiveEventStruct);

  // DEBUG
  STDAIR_LOG_DEBUG ("====================================================");
  STDAIR_LOG_DEBUG ("=       Beginning of the interactive session       =");
  STDAIR_LOG_DEBUG ("====================================================");

  // Initialise the GNU readline wrapper
  swift::SReadline lReader (lHistoryFilename, lHistorySize);
  initReadline (lReader);

  // Now we can ask user for a line
  std::string lUserInput;
  bool EndOfInput (false);
  Command_T::Type_T lCommandType (Command_T::NOP);

  while (lCommandType != Command_T::QUIT && EndOfInput == false) {

    // Update the interactive parameters which have not been updated yet
    lCurrentInteractiveDateTime = lCurrentInteractiveEventStruct.getEventTime ();
    lCurrentInteractiveEventType = lCurrentInteractiveEventStruct.getEventType ();

    // Prompt
    std::ostringstream oPromptStr;
    oPromptStr << "sevmgr "
               << stdair::EventType::getTypeLabelAsString(lCurrentInteractiveEventType)
               << " / " << lCurrentInteractiveDateTime << "> " ;
    // Call read-line, which will fill the list of tokens
    TokenList_T lTokenListByReadline;
    lUserInput = lReader.GetLine (oPromptStr.str(), lTokenListByReadline,
                                  EndOfInput);

    // The history can be saved to an arbitrary file at any time
    lReader.SaveHistory (lHistoryBackupFilename);

    // The end-of-input typically corresponds to a CTRL-D typed by the user
    if (EndOfInput) {
      std::cout << std::endl;
      break;
    }

    // Interpret the user input
    lCommandType = extractCommand (lTokenListByReadline);

    switch (lCommandType) {

      // ////////////////////////////// Help ////////////////////////
    case Command_T::HELP: {
      std::cout << std::endl;
      std::cout << "Commands: " << std::endl;
      std::cout << " help" << "\t\t" << "Display this help" << std::endl;
      std::cout << " quit" << "\t\t" << "Quit the application" << std::endl;
      std::cout << " list" << "\t\t" << "List events in the queue. It is "
                << "possible to filter events according to their types"
                << std::endl
                << "\t\t\t\t  'list_event BookingRequest'  "
                << "list all the booking requests" << std::endl
                << "\t\t\t\t  'list_event BreakPoint'  "
                << "list all the break points" << std::endl;
      std::cout << " select" << "\t\t"
                << "Select an event into the 'list' to become the current one. For instance, try the command:\n"
                << "\t\t  'select 2011-May-14 00:00:00'"
                << std::endl;
      std::cout << " display" << "\t"
                << "Display the current event" << std::endl;
      std::cout << " next" << "\t\t"
                << "Play the current event and pop the next one from the queue"
                << std::endl;
      std::cout << " run" << "\t\t"
                << "Play all the events until the next break-point, if any"
                << std::endl;
      std::cout << " \nDebug Commands" << std::endl;
      std::cout << " json_list" << "\t"
                << "List events in the queue in a JSON format"
                << std::endl;
      std::cout << " json_display" << "\t"
                << "Display the current event in a JSON format"
                << std::endl;
      std::cout << std::endl;
      break;
    }

      // ////////////////////////////// Quit ////////////////////////
    case Command_T::QUIT: {
      break;
    }

      // ////////////////////////////// List /////////////////////////
    case Command_T::LIST: {

      //
      std::ostringstream oEventListStr;

      if (lTokenListByReadline.empty() == true) {

        // If no parameter is given, list all the events in the queue
        oEventListStr << sevmgrService.list ();

      } else if (lTokenListByReadline.size() == 1) {

        assert (lTokenListByReadline.empty() == false);
        const std::string lEventTypeStr (lTokenListByReadline[0]);

        // If exactly one parameter is given, try to convert it into
        // an event type
        try {

          const stdair::EventType lEventType (lEventTypeStr);
          const stdair::EventType::EN_EventType& lActualEventType =
            lEventType.getType();
          oEventListStr << sevmgrService.list (lActualEventType);

        } catch (stdair::CodeConversionException e) {
          oEventListStr << "The event type '" << lEventTypeStr
                        << "' is not known. Try 'help' for "
                        << "more information on the 'list_event' command."
                        << std::endl;
        }
      } else {

        // If more than one parameter is given, display an error message
        oEventListStr << "The event type is not understood: try 'help' for "
                      << "more information on the 'list_event' command."
                      << std::endl;
      }
      std::cout << oEventListStr.str() << std::endl;
      STDAIR_LOG_DEBUG (oEventListStr.str());

      //
      break;
    }

      // ////////////////////////////// Select ////////////////////////
    case Command_T::SELECT: {

      //
      TokenList_T lTokenList = extractTokenListForDateTime (lTokenListByReadline)
;
      stdair::Date_T lUserDate = lCurrentInteractiveDateTime.date();
      stdair::Duration_T lUserTime = lCurrentInteractiveDateTime.time_of_day();
      parseEventDateTime (lTokenList, lUserDate, lUserTime);

      std::cout << "Try to select event: "
                << lUserDate << " " << lUserTime
                << std::endl;

      const stdair::DateTime_T lUserDateTime =
        boost::posix_time::ptime (lUserDate, lUserTime);

      const bool hasSelectBeenSuccessful =
        sevmgrService.select (lCurrentInteractiveEventStruct,
                              lUserDateTime);

      std::cout << "Selection successful: "
                << hasSelectBeenSuccessful << std::endl;

      //
      break;
    }

      // ////////////////////////////// Display ////////////////////////
    case Command_T::DISPLAY: {
      //
      std::cout << "Display" << std::endl;

      // DEBUG
      std::ostringstream oEventStr;
      oEventStr << lCurrentInteractiveEventStruct.describe();
      std::cout << oEventStr.str() << std::endl;
      STDAIR_LOG_DEBUG (oEventStr.str());

      //
      break;
    }

      // ////////////////////////////// Next ////////////////////////
    case Command_T::NEXT: {
      //
      std::cout << "Next" << std::endl;

      if (sevmgrService.isQueueDone() == true) {

        // DEBUG
        std::ostringstream oEmptyQueueStr;
        oEmptyQueueStr << "The event queue is empty: no event can be popped out."
;
        std::cout << oEmptyQueueStr.str() << std::endl;
        STDAIR_LOG_DEBUG (oEmptyQueueStr.str());

        //
        break;

      }

      // Get the next event from the event queue
      stdair::ProgressStatusSet lPPS =
        sevmgrService.popEvent (lCurrentInteractiveEventStruct);

      // DEBUG
      std::ostringstream oEventStr;
      oEventStr << "Poped event: '"
                << lCurrentInteractiveEventStruct.describe() << "'.";
      std::cout << oEventStr.str() << std::endl;
      STDAIR_LOG_DEBUG (oEventStr.str());

      //
      break;
    }

      // ////////////////////////////// Run ////////////////////////
    case Command_T::RUN: {
      //
      std::cout << "Run" << std::endl;

      // Delegate the call to the dedicated service
      sevmgrService.run (lCurrentInteractiveEventStruct);
      lCurrentInteractiveEventType = lCurrentInteractiveEventStruct.getEventType
();

      // DEBUG
      if (lCurrentInteractiveEventType == stdair::EventType::BRK_PT) {
        std::ostringstream oBreakPointStr;
        oBreakPointStr << "Break point found. Stop at: '"
                       << lCurrentInteractiveEventStruct.describe() << "'.";
        std::cout << oBreakPointStr.str() << std::endl;
        STDAIR_LOG_DEBUG (oBreakPointStr.str());
      } else {
        std::ostringstream oNoBreakPointStr;
        oNoBreakPointStr << "No break point found. All the events have been playe
d.\n"
                         << "The current event is the last one.";
        std::cout << oNoBreakPointStr.str() << std::endl;
        STDAIR_LOG_DEBUG (oNoBreakPointStr.str());
      }

      //
      break;
    }

      // ////////////////////////////// JSon List ////////////////////////

    case Command_T::JSON_LIST: {
      //
      std::cout << "JSON List" << std::endl;

      // Delegate the call to the dedicated service
      const std::string& lCSVEventQueueDumpAfter =
        sevmgrService.jsonExportEventQueue ();

      // DEBUG: Display the events queue JSON string
      std::cout << lCSVEventQueueDumpAfter << std::endl;
      STDAIR_LOG_DEBUG (lCSVEventQueueDumpAfter);

      break;
    }

      // ////////////////////////////// JSon Display ///////////////////////

    case Command_T::JSON_DISPLAY: {
      //
      std::cout << "JSON Display" << std::endl;

      // Delegate the call to the dedicated service
      const std::string& lCSVEventDumpAfter =
        sevmgrService.jsonExportEvent (lCurrentInteractiveEventStruct);

      // DEBUG: Display the event JSON string
      std::cout << lCSVEventDumpAfter << std::endl;
      STDAIR_LOG_DEBUG (lCSVEventDumpAfter);

      //
      break;
    }

      // ////////////////////////// Default / No value //////////////////////
    case Command_T::NOP: {
      break;
    }

    case Command_T::LAST_VALUE:
    default: {
      // DEBUG
      std::ostringstream oStr;
      oStr << "That command is not yet understood: '" << lUserInput
           << "' => " << lTokenListByReadline;
      STDAIR_LOG_DEBUG (oStr.str());
      std::cout << oStr.str() << std::endl;
    }
    }
  }

  // DEBUG
  STDAIR_LOG_DEBUG ("End of the session. Exiting.");
  std::cout << "End of the session. Exiting." << std::endl;

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

  /*
    Note: as that program is not intended to be run on a server in
    production, it is better not to catch the exceptions. When it
    happens (that an exception is throwned), that way we get the
    call stack.
  */

  return 0;
}
```

25.65 test/sevmgr/EventQueueManagementTestSuite.cpp File Reference

25.66 EventQueueManagementTestSuite.cpp

```

// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <map>
#include <cmath>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE EventQueueManagementTest
#include <boost/test/unit_test.hpp>
#include <boost/shared_ptr.hpp>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/service/Logger.hpp>
// SEvMgr
#include <sevmgr/SEVMGR_Service.hpp>
#include <sevmgr/config/sevmgr-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("EventQueueManagementTestSuite_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_tes
ts);
  }

  ~UnitTestConfig() {
  }
};

// Specific type definitions
typedef std::pair<stdair::Count_T, stdair::Count_T> NbOfEventsPair_T;
typedef std::map<const stdair::DemandStreamKeyStr_T,
                 NbOfEventsPair_T> NbOfEventsByDemandStreamMap_T;


// /////////////// 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 (sevmgr_simple_simulation_test) {

```

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

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

// Initialise the Sevmgr service object
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
SEVMGR::SEVMGR_Service sevmgrService (lLogParams);

const bool isQueueDone = sevmgrService.isQueueDone();
BOOST_REQUIRE_MESSAGE (isQueueDone == true,
                       "The event queue should be empty at this step. No "
                       << "insertion done.");

sevmgrService.buildSampleQueue ();

stdair::Count_T lNbOfEvents (sevmgrService.getQueueSize());

BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == false,
                       "The event queue should not be empty at this step. "
                       << "Two insertions done.");

stdair::Count_T idx = 1;
while (sevmgrService.isQueueDone() == false) {

  // Pop the next event out of the event queue
  stdair::EventStruct lEventStruct;
  const stdair::ProgressStatusSet lPPS =
    sevmgrService.popEvent (lEventStruct);

  // DEBUG
  STDAIR_LOG_DEBUG ("Poped event "<< idx << ": '"
                    << lEventStruct.describe() << "'.");
  STDAIR_LOG_DEBUG ("Progresss status: " << lPPS.describe());
  STDAIR_LOG_DEBUG ("Poped event: '"
                    << lEventStruct.describe() << "'.");

  // Iterate
  ++idx;
}

// Compensate for the last iteration
--idx;
// Compared the actual number of popped events with the expected one.
BOOST_REQUIRE_MESSAGE (idx == lNbOfEvents,
                       "Actual number of requests in the queue: "
                       << idx << ". Expected value: " << lNbOfEvents);

BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
                       "The event queue should be empty at this step: "
                       "the two events have been popped.");

STDAIR_LOG_DEBUG ("Re-added the events into the queue");

// Add again the four events into the queue thanks to
// sevmgrService.buildSampleQueue().
// Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
// Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
sevmgrService.buildSampleQueue ();

// Pop the next event out of the event queue
stdair::EventStruct lFirstEventStruct;
const stdair::ProgressStatusSet lFirstPS =
    sevmgrService.popEvent (lFirstEventStruct);

  // Extract the corresponding date
  const stdair::DateTime_T& lFirstEventDateTime =
    lFirstEventStruct.getEventTime ();
  const stdair::Date_T& lFirstRequestDate =
    lFirstEventDateTime.date();

  const stdair::Date_T lExpectedDate (2010, boost::gregorian::Jan, 21);
  BOOST_REQUIRE_MESSAGE (lFirstRequestDate == lExpectedDate,
                         "Date of the first event popped from the queue: "
                         << lFirstRequestDate << ". Should be: "
                         << lExpectedDate << " which is earlier in time.");

  STDAIR_LOG_DEBUG ("Reset the queue");
  sevmgrService.reset();

  BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
                         "The event queue has been reset: it should be empty "
                         << "at this step.");

  STDAIR_LOG_DEBUG ("Re-added the events into the queue one more time");

  // Add again the four events into the queue thanks to
  // sevmgrService.buildSampleQueue().
  // Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
  // Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
  sevmgrService.buildSampleQueue ();

  stdair::EventStruct lBreakPointStruct;
  sevmgrService.run(lBreakPointStruct);
  stdair::EventType::EN_EventType lBreakPointType =
    lBreakPointStruct.getEventType();

  BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
                         "The last event poppped from the queue should be a "
                         << "break point.");

  sevmgrService.run(lBreakPointStruct);
  lBreakPointType = lBreakPointStruct.getEventType();

  BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
                         "The last event poppped from the queue should be a "
                         << "break point.");

  // Extract the corresponding date
  const stdair::DateTime_T& lBPDateTime =
    lBreakPointStruct.getEventTime ();
  const stdair::Date_T& lBPDate =
    lBPDateTime.date();

  const stdair::Date_T lExpectedBPDate (2011, boost::gregorian::May, 14);
  BOOST_REQUIRE_MESSAGE (lBPDate == lExpectedBPDate,
                         "Date of the second break point popped from the queue: "
                         << lBPDate << ". Should be: "
                         << lExpectedBPDate << ".");

  // DEBUG
  STDAIR_LOG_DEBUG ("End of the simulation");

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

// End the test suite
BOOST_AUTO_TEST_SUITE_END()
```