

SEvMgr
1.00.0

Generated by Doxygen 1.8.1.1

Sun Dec 23 2012 20:13:18

Contents

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

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

10.6	M4 macro for the GNU Autotools	26
10.7	Using SEvMgr with dynamic linking	26
11	Test Rules	26
11.1	The Test File	26
11.2	The Reference File	26
11.3	Testing IT++ Library	26
12	Users Guide	27
12.1	Table of Contents	27
12.2	Introduction	27
12.3	Get Started	27
12.3.1	Get the SEvMgr library	27
12.3.2	Build the SEvMgr project	27
12.3.3	Build and Run the Tests	28
12.3.4	Install the SEvMgr Project (Binaries, Documentation)	28
12.4	Input file of SEvMgr Project	28
12.5	The schedule BOM Tree	29
12.5.1	Build of the schedule BOM tree	30
12.5.2	Display of the schedule BOM tree	30
12.6	Exploring the Predefined BOM Tree	73
12.6.1	Airline Network BOM Tree	73
12.6.2	Airline Schedule BOM Tree	73
12.7	Extending the BOM Tree	74
12.8	The travel solution calculation procedure	74
13	Supported Systems	74
13.1	Table of Contents	74
13.2	Introduction	74
14	SEvMgr Supported Systems (Previous Releases)	75
14.1	SEvMgr 3.9.1	75
14.2	SEvMgr 3.9.0	75
14.3	SEvMgr 3.8.1	75
15	Tutorials	75
15.1	Table of Contents	75
15.2	Preparing the AirSched Project for Development	75
15.3	Your first networkBuilde	75
15.3.1	Summary of the different steps	75
15.3.2	Result of the Batch Program	76
15.4	Network building with an input file	76

15.4.1	How to build a network input file?	76
15.4.2	Building the BOM tree with an input file	77
15.4.3	Result of the Batch Program	77
16	Command-Line Test to Demonstrate How To Use Sevmgr elements	77
17	Namespace Index	80
17.1	Namespace List	80
18	Class Index	80
18.1	Class Hierarchy	80
19	Class Index	81
19.1	Class List	81
20	File Index	82
20.1	File List	82
21	Namespace Documentation	83
21.1	bpt Namespace Reference	83
21.1.1	Typedef Documentation	83
21.2	SEVMGR Namespace Reference	83
21.2.1	Typedef Documentation	84
21.2.2	Function Documentation	86
21.2.3	Variable Documentation	86
21.3	stdair Namespace Reference	87
21.3.1	Detailed Description	87
22	Class Documentation	87
22.1	BomAbstract Class Reference	87
22.2	SEVMGR::BomJSONExport Class Reference	87
22.2.1	Detailed Description	87
22.2.2	Member Function Documentation	87
22.3	CmdAbstract Class Reference	88
22.4	SEVMGR::EventQueue Class Reference	88
22.4.1	Detailed Description	90
22.4.2	Member Typedef Documentation	90
22.4.3	Constructor & Destructor Documentation	91
22.4.4	Member Function Documentation	91
22.4.5	Friends And Related Function Documentation	97
22.4.6	Member Data Documentation	97
22.5	SEVMGR::EventQueueException Class Reference	98
22.5.1	Detailed Description	98

22.5.2	Constructor & Destructor Documentation	98
22.6	SEVMGR::EventQueueKey Struct Reference	99
22.6.1	Detailed Description	99
22.6.2	Constructor & Destructor Documentation	99
22.6.3	Member Function Documentation	99
22.7	SEVMGR::EventQueueManager Class Reference	100
22.7.1	Detailed Description	100
22.7.2	Friends And Related Function Documentation	101
22.8	FacServiceAbstract Class Reference	101
22.9	SEVMGR::FacSEVMGRServiceContext Class Reference	101
22.9.1	Detailed Description	101
22.9.2	Constructor & Destructor Documentation	102
22.9.3	Member Function Documentation	102
22.10	KeyAbstract Class Reference	102
22.11	SEVMGR::PYEventQueueManager Struct Reference	103
22.11.1	Detailed Description	103
22.11.2	Constructor & Destructor Documentation	103
22.11.3	Member Function Documentation	103
22.12	RootException Class Reference	104
22.13	ServiceAbstract Class Reference	104
22.14	SEVMGR::SEVMGR_Service Class Reference	104
22.14.1	Detailed Description	105
22.14.2	Constructor & Destructor Documentation	105
22.14.3	Member Function Documentation	106
22.15	SEVMGR::SEVMGR_ServiceContext Class Reference	113
22.15.1	Detailed Description	113
22.15.2	Friends And Related Function Documentation	113
22.16	SEVMGR::SEvMgrException Class Reference	114
22.16.1	Detailed Description	114
22.16.2	Constructor & Destructor Documentation	114
23	File Documentation	114
23.1	doc/local/authors.doc File Reference	114
23.2	doc/local/codingrules.doc File Reference	114
23.3	doc/local/copyright.doc File Reference	114
23.4	doc/local/documentation.doc File Reference	114
23.5	doc/local/features.doc File Reference	114
23.6	doc/local/help_wanted.doc File Reference	115
23.7	doc/local/howto_release.doc File Reference	115
23.8	doc/local/index.doc File Reference	115

23.9 doc/local/installation.doc File Reference	115
23.10 doc/local/linking.doc File Reference	115
23.11 doc/local/test.doc File Reference	115
23.12 doc/local/users_guide.doc File Reference	115
23.13 doc/local/verification.doc File Reference	115
23.14 doc/tutorial/tutorial.doc File Reference	115
23.15 sevmgr/basic/BasConst.cpp File Reference	115
23.16 BasConst.cpp	115
23.17 sevmgr/basic/BasConst_EventQueueManager.hpp File Reference	115
23.18 BasConst_EventQueueManager.hpp	116
23.19 sevmgr/basic/BasConst_SEVMGR_Service.hpp File Reference	116
23.20 BasConst_SEVMGR_Service.hpp	116
23.21 sevmgr/basic/BasParserTypes.hpp File Reference	116
23.22 BasParserTypes.hpp	118
23.23 sevmgr/batches/sevmgr_demo.cpp File Reference	119
23.23.1 Function Documentation	119
23.23.2 Variable Documentation	119
23.24 sevmgr_demo.cpp	120
23.25 sevmgr/bom/BomJSONExport.cpp File Reference	122
23.26 BomJSONExport.cpp	122
23.27 sevmgr/bom/BomJSONExport.hpp File Reference	123
23.28 BomJSONExport.hpp	123
23.29 sevmgr/bom/EventQueue.cpp File Reference	124
23.30 EventQueue.cpp	124
23.31 sevmgr/bom/EventQueue.hpp File Reference	130
23.32 EventQueue.hpp	130
23.33 sevmgr/bom/EventQueueKey.cpp File Reference	133
23.34 EventQueueKey.cpp	133
23.35 sevmgr/bom/EventQueueKey.hpp File Reference	134
23.36 EventQueueKey.hpp	134
23.37 sevmgr/bom/EventQueueTypes.hpp File Reference	135
23.38 EventQueueTypes.hpp	135
23.39 sevmgr/command/EventQueueManager.cpp File Reference	135
23.40 EventQueueManager.cpp	136
23.41 sevmgr/command/EventQueueManager.hpp File Reference	140
23.42 EventQueueManager.hpp	140
23.43 sevmgr/config/sevmgr-paths.hpp File Reference	142
23.43.1 Macro Definition Documentation	142
23.44 sevmgr-paths.hpp	143
23.45 sevmgr/config/sevmgr-paths.hpp.in File Reference	144

23.45.1 Macro Definition Documentation	144
23.46sevmgr-paths.hpp.in	146
23.47sevmgr/factory/FacSEVMGRServiceContext.cpp File Reference	146
23.48FacSEVMGRServiceContext.cpp	146
23.49sevmgr/factory/FacSEVMGRServiceContext.hpp File Reference	147
23.50FacSEVMGRServiceContext.hpp	147
23.51sevmgr/python/pysevmgr.cpp File Reference	148
23.51.1 Function Documentation	148
23.52pysevmgr.cpp	148
23.53sevmgr/service/SEVMGR_Service.cpp File Reference	150
23.54SEVMGR_Service.cpp	151
23.55sevmgr/service/SEVMGR_ServiceContext.cpp File Reference	160
23.56SEVMGR_ServiceContext.cpp	160
23.57sevmgr/service/SEVMGR_ServiceContext.hpp File Reference	162
23.58SEVMGR_ServiceContext.hpp	162
23.59sevmgr/SEVMGR_Exceptions.hpp File Reference	163
23.60SEVMGR_Exceptions.hpp	163
23.61sevmgr/SEVMGR_Service.hpp File Reference	164
23.62SEVMGR_Service.hpp	164
23.63sevmgr/SEVMGR_Types.hpp File Reference	166
23.64SEVMGR_Types.hpp	167
23.65sevmgr/ui/cmdline/sevmgr.cpp File Reference	167
23.66sevmgr.cpp	167
23.67test/sevmgr/EventQueueManagementTestSuite.cpp File Reference	177
23.68EventQueueManagementTestSuite.cpp	177

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](#)
- [SOI \(C++ DB API\)](#)

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

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

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

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

4.2 Preamble

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

to use the modified definitions.)

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

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

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

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

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

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

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

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

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

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

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

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

1. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise)

that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

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

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

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

1. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
1. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

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

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

4.3.1 NO WARRANTY

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

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

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

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

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

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

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

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

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

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

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

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

That's all there is to it!

Source

5 Documentation Rules

5.1 General Rules

All classes in 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

- [Fedora/RedHat Linux distributions](#)
- [SEVMgr Requirements](#)
- [Basic Installation](#)

- [Compilers and Options](#)
- [Compiling For Multiple Architectures](#)
- [Installation Names](#)
- [Optional Features](#)
- [Particular systems](#)
- [Specifying the System Type](#)
- [Sharing Defaults](#)
- [Defining Variables](#)
- [‘cmake’ Invocation](#)

9.2 Fedora/RedHat Linux distributions

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

```
yum -y install 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``)
- [SOXI](#) - 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 output 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_AIRAC_PREFIX=${INSTALL_BASEDIR}/airrac-stable \
  -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/rmol-stable \
  -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ${LIBSUFFIX_4_CMAKE} ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
```

```

-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: 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;rmol;sevmgrlib
-- Binaries to build/install ..... : airrac;rmol;sevmgr_parseInventory;sevmgr
-- Modules to test ..... : sevmgr
-- Binaries to test ..... : InventoryTestSuite
--
-- * Module ..... : sevmgr
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib;rmol;sevmgrlib
--   + Libraries to build/install . : sevmgrlib
--   + Executables to build/install : sevmgr_parseInventory;sevmgr
--   + Tests to perform ..... : InventoryTestSuite
--
-- 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;unit_
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_rege
--
-- * Readline:
--   - READLINE_VERSION ..... : 6.2
--   - READLINE_INCLUDE_DIR ..... : /usr/include
--   - READLINE_LIBRARY ..... : /usr/lib64/libreadline.so
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ..... : 0.38.0
--   - STDAIR_BINARY_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/bin
--   - STDAIR_EXECUTABLES ..... : stdair
--   - STDAIR_LIBRARY_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/lib64
--   - STDAIR_LIBRARIES ..... : stdairlib;stdairuiclib
--   - STDAIR_INCLUDE_DIRS ..... : /home/dan/dev/deliveries/stdair-0.38.0/include
--   - STDAIR_SAMPLE_DIR ..... : /home/dan/dev/deliveries/stdair-0.38.0/share/stdair/samples

```

```
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/dan/dev/sim/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

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

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'`).

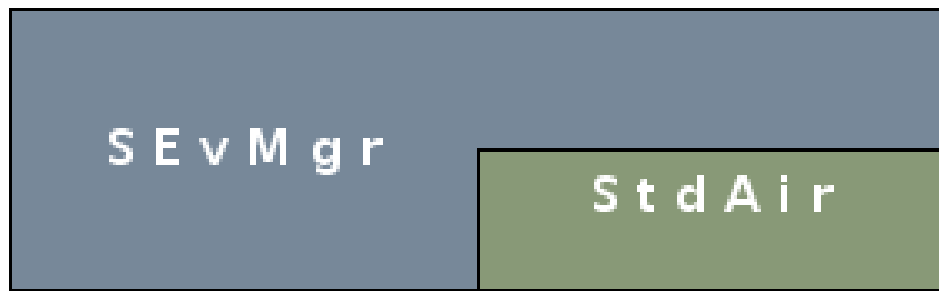


Figure 1: SEvMgr Dependencies

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 the `'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

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

12.2 Introduction

The `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 SEV-MGR: :FlightPeriodStruct). The fields are as follows:

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

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

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

12.5 The schedule BOM Tree

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

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

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



```

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

```

```

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

```

```

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

```

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

```

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

```

```
SQL1 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,
SQL1 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,
SQL1 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: SQL1, 2010-Feb-01
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQL1 2010-Feb-01, SIN-BKK 2010-Feb-01, 08:20:00, 2010-Feb-01, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL1 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,
SQL1 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, 0, 0, 0, 0, 300, 0,
SQL1 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,
SQL1 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,
SQL1 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: SQL1, 2010-Feb-02
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
  Elapsed, Distance, Capacity,
SQL1 2010-Feb-02, SIN-BKK 2010-Feb-02, 08:20:00, 2010-Feb-02, 11:00:00, 07:40:
  00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
  CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL1 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,
SQL1 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, 0, 0, 0, 0, 300, 0,
SQL1 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, 0, 0, 0, 0, 300, 0,
*****
*****
Subclasses:
-----
```

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

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



```

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

```

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

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

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

```

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

```

```

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

```

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

```

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

```



```

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

```

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

```

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

```

```
SQL2 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.63638e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL2 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Jan-17
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL2 2010-Jan-17, SIN-HND, 2010-Jan-17, 09:20:00, 2010-Jan-17, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.39291e-319, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL2 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Jan-18
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL2 2010-Jan-18, SIN-HND, 2010-Jan-18, 09:20:00, 2010-Jan-18, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhycAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 2.14469e-319, 0, 0, 0, 0,
*****
*****
Buckets:
```

```

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

```

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

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

```
SQL2 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,
SQL2 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: SQL2, 2010-Jan-25
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL2 2010-Jan-25, SIN-HND, 2010-Jan-25, 09:20:00, 2010-Jan-25, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 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,
SQL2 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 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: SQL2, 2010-Jan-26
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
Elapsed, Distance, Capacity,
SQL2 2010-Jan-26, SIN-HND, 2010-Jan-26, 09:20:00, 2010-Jan-26, 12:00:00, 07:40:
00, 0, -05:00:00, 6300, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQL2 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 200
, 9, 0, 0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQL2 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQL2 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,
SQL2 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,
SQL2 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
0, 0, 0, 0, 0,
*****
```



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

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

```

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

```

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

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

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

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

```

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

```



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

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

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

```

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

```

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

```

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

```

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

```

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

```


12.6 Exploring the Predefined BOM Tree

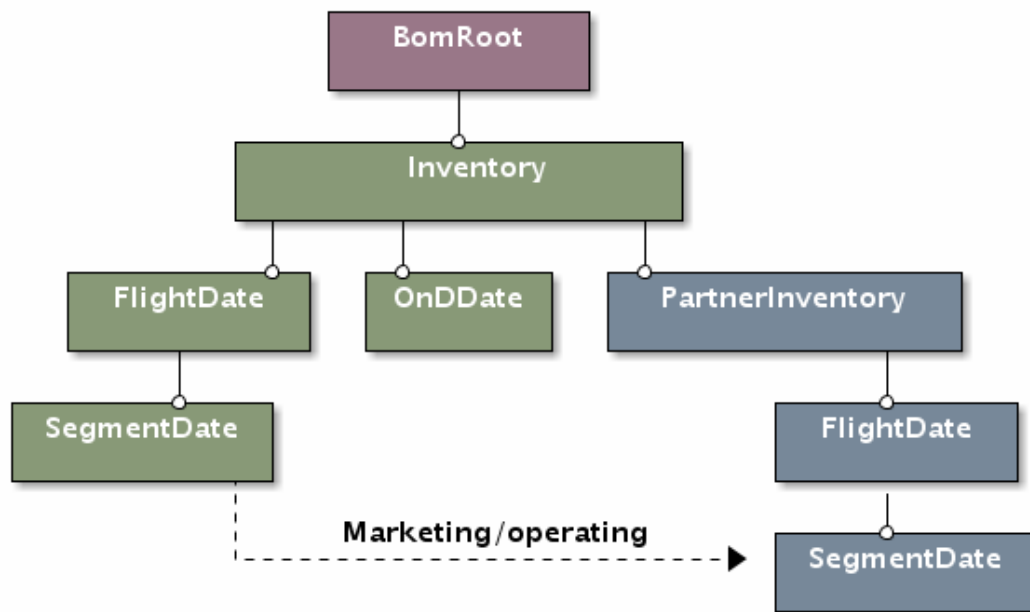


Figure 2: SEvMgr BOM tree

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

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

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 successful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

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

If you manage to compile, install and test the 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

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

15.2 Preparing the AirSched Project for Development

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

15.3 Your first networkBuilde

15.3.1 Summary of the different steps

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

First, we instantiate the `AIRSCHEd_Service` object:

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

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

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

15.3.2 Result of the Batch Program

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

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

and after the network building:

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

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

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

15.4 Network building with an input file

15.4.1 How to build a network input file?

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

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

We deduce:

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

We can deduce a part of our network rule file :

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

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

The input file is now:

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

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

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

We obtain:

15.4.2 Building the BOM tree with an input file

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

15.4.3 Result of the Batch Program

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

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

the last lines of the log output should look like:

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

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

16 Command-Line Test to Demonstrate How To 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_tests);
    }

    ~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 (UnitTestFixture);

// 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 ("Popped event " << idx << ": ' "
                        << lEventStruct.describe() << "'.");
        STDAIR_LOG_DEBUG ("Progressss status: " << lPPS.describe());
        STDAIR_LOG_DEBUG ("Popped 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 popped 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 popped 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 Namespace Index

17.1 Namespace List

Here is a list of all namespaces with brief descriptions:

bpt	83
SEVMGR	83
stdair	
Forward declarations	87

18 Class Index

18.1 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 >
std::basic_ifstream< wchar_t >
std::basic_ios< char >
std::basic_ios< wchar_t >
std::basic_iostream< char >
std::basic_iostream< wchar_t >
std::basic_istream< char >
std::basic_istream< wchar_t >
std::basic_istreamstream< char >
std::basic_istreamstream< wchar_t >
std::basic_ofstream< char >
std::basic_ofstream< wchar_t >
std::basic_ostream< char >
std::basic_ostream< wchar_t >
std::basic_ostreamstream< char >
std::basic_ostreamstream< wchar_t >
std::basic_string< char >
std::basic_string< wchar_t >
std::basic_stringstream< char >
std::basic_stringstream< wchar_t >
```

BomAbstract	87
SEVMGR::EventQueue	88
SEVMGR::BomJSONExport	87
CmdAbstract	88
SEVMGR::EventQueueManager	100
FacServiceAbstract	101
SEVMGR::FacSEVMGRServiceContext	101
KeyAbstract	102
SEVMGR::EventQueueKey	99
SEVMGR::PYEventQueueManager	103
RootException	104
SEVMGR::SEvMgrException	114
SEVMGR::EventQueueException	98
ServiceAbstract	104
SEVMGR::SEVMGR_ServiceContext	113
SEVMGR::SEVMGR_Service	104

19 Class Index

19.1 Class List

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

BomAbstract	87
SEVMGR::BomJSONExport Utility class to export StdAir objects in a JSON format	87
CmdAbstract	88
SEVMGR::EventQueue Class holding event structures	88
SEVMGR::EventQueueException	98
SEVMGR::EventQueueKey	99
SEVMGR::EventQueueManager Utility class for Demand and DemandStream objects	100
FacServiceAbstract	101
SEVMGR::FacSEVMGRServiceContext	101
KeyAbstract	102
SEVMGR::PYEventQueueManager	103
RootException	104
ServiceAbstract	104
SEVMGR::SEVMGR_Service Class holding the services related to Travel Demand Generation	104
SEVMGR::SEVMGR_ServiceContext Class holding the context of the Sevmgr services	113
SEVMGR::SEvMgrException	114

20 File Index

20.1 File List

Here is a list of all files with brief descriptions:

sevmgr/SEVMGR_Exceptions.hpp	163
sevmgr/SEVMGR_Service.hpp	164
sevmgr/SEVMGR_Types.hpp	167
sevmgr/basic/BasConst.cpp	115
sevmgr/basic/BasConst_EventQueueManager.hpp	116

sevmgr/basic/ BasConst_SEVMGR_Service.hpp	116
sevmgr/basic/ BasParserTypes.hpp	118
sevmgr/batches/ sevmgr_demo.cpp	120
sevmgr/bom/ BomJSONExport.cpp	122
sevmgr/bom/ BomJSONExport.hpp	123
sevmgr/bom/ EventQueue.cpp	124
sevmgr/bom/ EventQueue.hpp	130
sevmgr/bom/ EventQueueKey.cpp	133
sevmgr/bom/ EventQueueKey.hpp	134
sevmgr/bom/ EventQueueTypes.hpp	135
sevmgr/command/ EventQueueManager.cpp	136
sevmgr/command/ EventQueueManager.hpp	140
sevmgr/config/ sevmgr-paths.hpp	143
sevmgr/config/ sevmgr-paths.hpp.in	146
sevmgr/factory/ FacSEVMGRServiceContext.cpp	146
sevmgr/factory/ FacSEVMGRServiceContext.hpp	147
sevmgr/python/ pysevmgr.cpp	148
sevmgr/service/ SEVMGR_Service.cpp	151
sevmgr/service/ SEVMGR_ServiceContext.cpp	160
sevmgr/service/ SEVMGR_ServiceContext.hpp	162
sevmgr/ui/cmdline/ sevmgr.cpp	167
test/sevmgr/ EventQueueManagementTestSuite.cpp	177

21 Namespace Documentation

21.1 bpt Namespace Reference

Typedefs

- typedef char [ptree](#)

21.1.1 Typedef Documentation

21.1.1.1 typedef char bpt::ptree

Definition at line 24 of file [BomJSONExport.cpp](#).

21.2 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](#)

21.2.1 Typedef Documentation

21.2.1.1 typedef char [SEVMGR::char_t](#)

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

21.2.1.2 typedef boost::spirit::classic::file_iterator< [char_t](#) > [SEVMGR::iterator_t](#)

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

21.2.1.3 typedef boost::spirit::classic::scanner< [iterator_t](#) > [SEVMGR::scanner_t](#)

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

21.2.1.4 typedef boost::spirit::classic::rule< [scanner_t](#) > [SEVMGR::rule_t](#)

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

21.2.1.5 `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](#).

21.2.1.6 `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](#).

21.2.1.7 `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](#).

21.2.1.8 `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](#).

21.2.1.9 `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](#).

21.2.1.10 `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](#).

21.2.1.11 `typedef boost::spirit::classic::chset<char_t> SEVMGR::chset_t`

character set

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

21.2.1.12 `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](#).

21.2.1.13 `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](#).

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

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

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

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

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

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

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

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

21.2.1.18 `typedef std::list<EventQueue*> SEVMGR::EventQueueList_T`

Define the [EventQueue](#) list.

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

21.2.1.19 `typedef std::map<const stdair::MapKey_T, EventQueue*> SEVMGR::EventQueueMap_T`

Define the [EventQueue](#) map.

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

21.2.1.20 `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](#).

21.2.1.21 `typedef std::string SEVMGR::EventQueueID_T`

Define an ID for an [EventQueue](#) object.

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

21.2.1.22 `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](#).

21.2.2 Function Documentation

21.2.2.1 `const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID ("EQ01")`

Default name for the [SEVMGR_Service](#). Default ID for the event queue.

21.2.3 Variable Documentation

21.2.3.1 `const EventQueueID_T SEVMGR::DEFAULT_EVENT_QUEUE_ID`

Default ID for the event queue.

21.3 stdair Namespace Reference

Forward declarations.

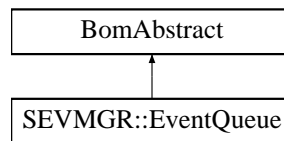
21.3.1 Detailed Description

Forward declarations.

22 Class Documentation

22.1 BomAbstract Class Reference

Inheritance diagram for BomAbstract:



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

- sevmgr/bom/[EventQueue.hpp](#)

22.2 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 &)

22.2.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

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

22.2.2 Member Function Documentation

22.2.2.1 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

<i>std::ostream&</i>	Output stream in which the events should be logged/dumped.
<i>const</i> EventQueue &	Events queue to be stored in JSON-ified format.
<i>const</i> stdair::EventType::EN_EventType&	Filter to select objects with a certain event type.

Definition at line 32 of file [BomJSONExport.cpp](#).

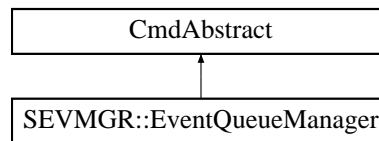
References [SEVMGR::EventQueue::getEventList\(\)](#).

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

- sevmgr/bom/[BomJSONExport.hpp](#)
- sevmgr/bom/[BomJSONExport.cpp](#)

22.3 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



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

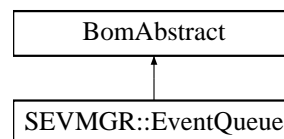
- sevmgr/command/[EventQueueManager.hpp](#)

22.4 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 &iProgress-Status)
- 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 &iProgress-Status)
- void [updateStatus](#) (const stdair::EventType::EN_EventType &, const stdair::NbOfEvents_T &iActualTotalNb-OfEvents)
- 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](#)

22.4.1 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](#).

22.4.2 Member Typedef Documentation

22.4.2.1 typedef EventQueueKey SEVMGR::EventQueue::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 77 of file [EventQueue.hpp](#).

22.4.3 Constructor & Destructor Documentation

22.4.3.1 SEVMGR::EventQueue::EventQueue (const Key_T & iKey) [protected]

Constructor.

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

22.4.3.2 SEVMGR::EventQueue::EventQueue (const EventQueue & iEventQueue) [protected]

Default copy constructor.

Definition at line 32 of file [EventQueue.cpp](#).

22.4.3.3 SEVMGR::EventQueue::~~EventQueue () [protected]

Destructor.

Definition at line 40 of file [EventQueue.cpp](#).

References [_eventList](#).

22.4.4 Member Function Documentation

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

Get the event queue key.

Definition at line 83 of file [EventQueue.hpp](#).

References [_key](#).

22.4.4.2 `BomAbstract* const SEVMGR::EventQueue::getParent () const` `[inline]`

Get the parent object.

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

References [_parent](#).

22.4.4.3 `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\(\)](#).

22.4.4.4 `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](#).

22.4.4.5 `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\(\)](#).

22.4.4.6 `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](#).

22.4.4.7 `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](#).

22.4.4.8 `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](#).

22.4.4.9 `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 324 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), and [display\(\)](#).

22.4.4.10 `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\(\)](#).

22.4.4.11 `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\(\)](#).

22.4.4.12 `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\(\)](#).

22.4.4.13 `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\(\)](#).

22.4.4.14 `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\(\)](#).

22.4.4.15 `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](#).

22.4.4.16 `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](#).

22.4.4.17 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](#).

22.4.4.18 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](#).

22.4.4.19 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 308 of file [EventQueue.cpp](#).

References [_progressStatusMap](#).

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

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 182 of file [EventQueue.hpp](#).

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

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

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Definition at line 191 of file [EventQueue.hpp](#).

22.4.4.22 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\(\)](#).

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

Get the event list description.

Definition at line 64 of file [EventQueue.cpp](#).

References [_eventList](#), [describeKey\(\)](#), and [toString\(\)](#).

22.4.4.24 `std::string SEVMGR::EventQueue::list (const stdair::EventType::EN_EventType & iType) const`

Get the event list description for a given event type

Definition at line 82 of file [EventQueue.cpp](#).

References [_eventList](#), [describeKey\(\)](#), and [toString\(\)](#).

22.4.4.25 `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\(\)](#).

22.4.4.26 `std::string SEVMGR::EventQueue::display () const`

Definition at line 55 of file [EventQueue.cpp](#).

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

Referenced by [calculateProgress\(\)](#), [getActualTotalNbOfEvents\(\)](#), [getCurrentNbOfEvents\(\)](#), [getExpectedTotalNbOfEvents\(\)](#), [getStatus\(\)](#), and [hasProgressStatus\(\)](#).

22.4.4.27 `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](#).

22.4.4.28 `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 365 of file [EventQueue.cpp](#).

References [_eventList](#), [_progressStatus](#), [describeKey\(\)](#), [getStatus\(\)](#), and [setStatus\(\)](#).

22.4.4.29 `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

<i>stdair::EventStruct&</i>	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 433 of file [EventQueue.cpp](#).

References [_eventList](#).

22.4.4.30 `bool SEVMGR::EventQueue::hasEventDateTime (const stdair::DateTime_T & iDateTime)`

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

Definition at line 466 of file [EventQueue.cpp](#).

References [_eventList](#).

22.4.4.31 `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\(\)](#).

22.4.4.32 `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.

Definition at line 257 of file [EventQueue.cpp](#).

References [_progressStatus](#), and [updateStatus\(\)](#).

22.4.4.33 `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\(\)](#).

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

22.4.4.34 `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](#).

22.4.4.35 `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](#).

22.4.4.36 `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 347 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), and [display\(\)](#).

22.4.4.37 `stdair::Count.T SEVMGR::EventQueue::getQueueSize () const`

Queue size

Definition at line 102 of file [EventQueue.cpp](#).

References [_eventList](#).

22.4.4.38 `bool SEVMGR::EventQueue::isQueueEmpty () const`

Is queue empty

Definition at line 107 of file [EventQueue.cpp](#).

References [_eventList](#).

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

22.4.5 Friends And Related Function Documentation

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

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

22.4.5.2 friend class stdair::FacBomManager [friend]

Definition at line 70 of file [EventQueue.hpp](#).

22.4.6 Member Data Documentation

22.4.6.1 Key_T SEVMGR::EventQueue::_key [protected]

Primary key (ID).

Definition at line 382 of file [EventQueue.hpp](#).

Referenced by [describeKey\(\)](#), and [getKey\(\)](#).

22.4.6.2 BomAbstract* SEVMGR::EventQueue::_parent [protected]

Pointer on the parent class (BomRoot).

Definition at line 387 of file [EventQueue.hpp](#).

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

22.4.6.3 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\(\)](#).

22.4.6.4 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\(\)](#).

22.4.6.5 stdair::ProgressStatus SEVMGR::EventQueue::_progressStatus [protected]

Counters holding the overall progress status.

Definition at line 404 of file [EventQueue.hpp](#).

Referenced by [addStatus\(\)](#), [calculateProgress\(\)](#), [getActualTotalNbOfEvents\(\)](#), [getCurrentNbOfEvents\(\)](#), [getExpectedTotalNbOfEvents\(\)](#), [getStatus\(\)](#), [popEvent\(\)](#), [reset\(\)](#), [setCurrentNbOfEvents\(\)](#), [setExpectedTotalNbOfEvents\(\)](#), [setStatus\(\)](#), [toString\(\)](#), and [updateStatus\(\)](#).

22.4.6.6 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\(\)](#).

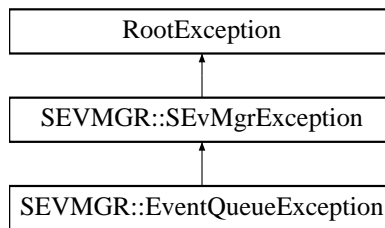
The documentation for this class was generated from the following files:

- [sevmgr/bom/EventQueue.hpp](#)
- [sevmgr/bom/EventQueue.cpp](#)

22.5 SEVMGR::EventQueueException Class Reference

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

Inheritance diagram for SEVMGR::EventQueueException:



Public Member Functions

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

22.5.1 Detailed Description

[EventQueue](#).

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

22.5.2 Constructor & Destructor Documentation

22.5.2.1 SEVMGR::EventQueueException::EventQueueException (const std::string &iWhat) [inline]

Constructor.

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

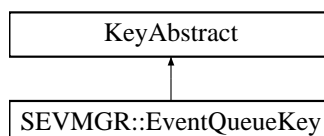
The documentation for this class was generated from the following file:

- [sevmgr/SEVMGR_Exceptions.hpp](#)

22.6 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 &ioln)
- const std::string [toString](#) () const

22.6.1 Detailed Description

Key of eventqueue.

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

22.6.2 Constructor & Destructor Documentation

22.6.2.1 SEVMGR::EventQueueKey::EventQueueKey (const EventQueueID_T & iEventQueueID)

Constructors.

Definition at line 12 of file [EventQueueKey.cpp](#).

22.6.2.2 SEVMGR::EventQueueKey::EventQueueKey (const EventQueueKey & iKey)

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

22.6.2.3 SEVMGR::EventQueueKey::~~EventQueueKey ()

Destructor.

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

22.6.3 Member Function Documentation

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

Get the ID of the [EventQueue](#) object.

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

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

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

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

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

22.6.3.3 void SEVMGR::EventQueueKey::fromStream (std::istream & ioln)

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

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

22.6.3.4 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:

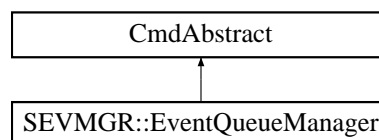
- [sevmgr/bom/EventQueueKey.hpp](#)
- [sevmgr/bom/EventQueueKey.cpp](#)

22.7 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](#)

22.7.1 Detailed Description

Utility class for Demand and DemandStream objects.

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

22.7.2 Friends And Related Function Documentation

22.7.2.1 friend class SEVMGR_Service [friend]

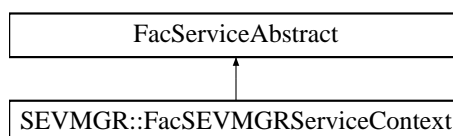
Definition at line 28 of file [EventQueueManager.hpp](#).

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

- [sevmgr/command/EventQueueManager.hpp](#)
- [sevmgr/command/EventQueueManager.cpp](#)

22.8 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



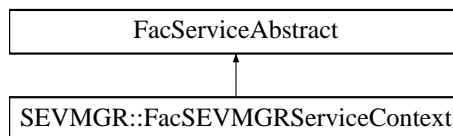
The documentation for this class was generated from the following file:

- [sevmgr/factory/FacSEVMGRServiceContext.hpp](#)

22.9 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\(\)](#)

22.9.1 Detailed Description

Factory for Bucket.

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

22.9.2 Constructor & Destructor Documentation

22.9.2.1 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](#).

22.9.2.2 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\(\)](#).

22.9.3 Member Function Documentation

22.9.3.1 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\(\)](#).

22.9.3.2 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.

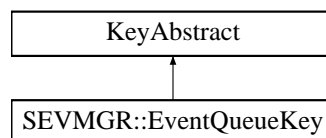
Definition at line 34 of file [FacSEVMGRServiceContext.cpp](#).

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

- [sevmgr/factory/FacSEVMGRServiceContext.hpp](#)
- [sevmgr/factory/FacSEVMGRServiceContext.cpp](#)

22.10 KeyAbstract Class Reference

Inheritance diagram for KeyAbstract:



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

- [sevmgr/bom/EventQueueKey.hpp](#)

22.11 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)

22.11.1 Detailed Description

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

22.11.2 Constructor & Destructor Documentation

22.11.2.1 SEVMGR::PYEventQueueManager::PYEventQueueManager () [inline]

Default constructor.

Definition at line 76 of file [pysevmgr.cpp](#).

22.11.2.2 SEVMGR::PYEventQueueManager::PYEventQueueManager (const PYEventQueueManager & iPYEventQueueManager) [inline]

Default copy constructor.

Definition at line 80 of file [pysevmgr.cpp](#).

22.11.2.3 SEVMGR::PYEventQueueManager::~~PYEventQueueManager () [inline]

Default constructor.

Definition at line 86 of file [pysevmgr.cpp](#).

22.11.3 Member Function Documentation

22.11.3.1 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\(\)](#).

22.11.3.2 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](#).

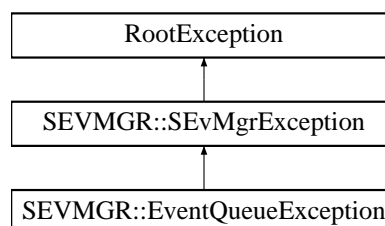
Referenced by [BOOST_PYTHON_MODULE\(\)](#).

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

- [sevmgr/python/pysevmgr.cpp](#)

22.12 RootException Class Reference

Inheritance diagram for RootException:

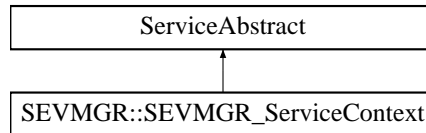


The documentation for this class was generated from the following file:

- [sevmgr/SEVMGR_Exceptions.hpp](#)

22.13 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



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

- [sevmgr/service/SEVMGR_ServiceContext.hpp](#)

22.14 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

22.14.1 Detailed Description

class holding the services related to Travel Demand Generation.

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

22.14.2 Constructor & Destructor Documentation

22.14.2.1 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

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

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

22.14.2.2 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

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

Definition at line 64 of file [SEVMGR_Service.cpp](#).

22.14.2.3 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

<i>stdair::STDAIR_ServicePtr_T</i>	Handler on the <code>STDAIR_Service</code> .
------------------------------------	----------------------------------------------

Definition at line 85 of file [SEVMGR_Service.cpp](#).

22.14.2.4 SEVMGR::SEVMGR_Service::~~SEVMGR_Service ()

Destructor.

Definition at line 101 of file [SEVMGR_Service.cpp](#).

22.14.3 Member Function Documentation

22.14.3.1 void SEVMGR::SEVMGR_Service::buildSampleQueue ()

Build a sample event queue.

Definition at line 175 of file [SEVMGR_Service.cpp](#).

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

22.14.3.2 *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

<i>const</i>	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](#).

22.14.3.3 `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\(\)](#).

22.14.3.4 `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](#).

22.14.3.5 `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

<i>const</i>	<code>stdair::DateTime_T</code> 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](#).

22.14.3.6 `template<class EventGenerator > void SEVMGR::SEVMGR_Service::addEventGenerator (EventGenerator & iEventGenerator) const`

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 appellant according the EventGenerator type.

Note

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

22.14.3.7 `void SEVMGR::SEVMGR_Service::addEvent (stdair::EventStruct & iEventStruct) const`

Add an event to the queue.

Definition at line 596 of file [SEVMGR_Service.cpp](#).

22.14.3.8 `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](#).

22.14.3.9 `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

<i>const</i>	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](#).

22.14.3.10 `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

<i>const</i>	stdair::EventType::EN_EventType& Type of the events for which the actual total count is updated.
--------------	--------------------------------------------------------------------------------------------------

Returns

const std::pair<Count_T, Expected Actual count of such events already generated

Definition at line 478 of file [SEVMGR_Service.cpp](#).

22.14.3.11 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\(\)](#).

22.14.3.12 bool SEVMGR::SEVMGR_Service::hasProgressStatus (const std::pair<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](#).

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

Definition at line 580 of file [SEVMGR_Service.cpp](#).

22.14.3.14 const std::pair<Count_T & SEVMGR::SEVMGR_Service::getQueueSize () const

Get the size of the queue.

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

22.14.3.15 template<class EventGenerator , class Key > EventGenerator& SEVMGR::SEVMGR_Service::getEventGenerator (const Key & iKey) const

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 appellant according the EventGenerator type.

Note

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

22.14.3.16 template<class EventGenerator , class Key > bool SEVMGR::SEVMGR_Service::hasEventGenerator (const Key & iKey) const

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

Be careful, this method is not implemented: its implementation is left to the appellant according the EventGenerator type.

Note

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

22.14.3.17 template<class EventGenerator > const std::list<EventGenerator*> SEVMGR::SEVMGR_Service::getEventGeneratorList () const

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 appellant according the EventGenerator type.

Note

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

22.14.3.18 `template<class EventGenerator > bool SEVMGR::SEVMGR_Service::hasEventGeneratorList () const`

Check whether there are event generator objects.

Be careful, this method is not implemented: its implementation is left to the appellant according the EventGenerator type.

Note

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

22.14.3.19 `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\(\)](#).

22.14.3.20 `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

<i>const</i>	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\(\)](#).

22.14.3.21 `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\(\)](#).

22.14.3.22 `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

<i>const</i>	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\(\)](#).

22.14.3.23 `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\(\)](#).

22.14.3.24 `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\(\)](#).

22.14.3.25 `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](#).

22.14.3.26 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\(\)](#).

22.14.3.27 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

<i>const</i>	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\(\)](#).

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

Dispatch the JSon command string to the corresponding service.

Parameters

<i>const</i>	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\(\)](#).

22.14.3.29 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\(\)](#).

22.14.3.30 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:

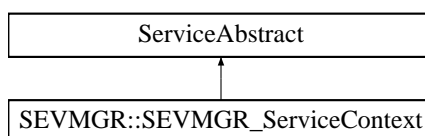
- [sevmgr/SEVMGR_Service.hpp](#)
- [sevmgr/service/SEVMGR_Service.cpp](#)

22.15 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](#)

22.15.1 Detailed Description

Class holding the context of the Sevmgr services.

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

22.15.2 Friends And Related Function Documentation

22.15.2.1 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](#).

22.15.2.2 friend class FacSEVMGRServiceContext [friend]

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

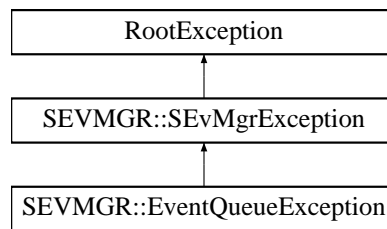
The documentation for this class was generated from the following files:

- [sevmgr/service/SEVMGR_ServiceContext.hpp](#)
- [sevmgr/service/SEVMGR_ServiceContext.cpp](#)

22.16 SEVMGR::SEvMgrException Class Reference

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

Inheritance diagram for SEVMGR::SEvMgrException:



Public Member Functions

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

22.16.1 Detailed Description

Root exception for the Sevmgr component

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

22.16.2 Constructor & Destructor Documentation

22.16.2.1 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](#)

23 File Documentation

23.1 doc/local/authors.doc File Reference

23.2 doc/local/codingrules.doc File Reference

23.3 doc/local/copyright.doc File Reference

23.4 doc/local/documentation.doc File Reference

23.5 doc/local/features.doc File Reference

23.6 doc/local/help_wanted.doc File Reference

23.7 doc/local/howto_release.doc File Reference

23.8 doc/local/index.doc File Reference

23.9 doc/local/installation.doc File Reference

23.10 doc/local/linking.doc File Reference

23.11 doc/local/test.doc File Reference

23.12 doc/local/users_guide.doc File Reference

23.13 doc/local/verification.doc File Reference

23.14 doc/tutorial/tutorial.doc File Reference

23.15 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")

23.16 BasConst.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/BasConst_General.hpp>
00006 // Sevmgr
00007 #include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
00008 #include <sevmgr/basic/BasConst_EventQueueManager.hpp>
00009
00010 namespace SEVMGR {
00011
00013     // const std::string DEFAULT_SEVMGR_SERVICE_NAME = "sevmgr";
00014
00016     const EventQueueID_T DEFAULT_EVENT_QUEUE_ID
00017     ("EQ01");
00018 }
```

23.17 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](#)

23.18 BasConst_EventQueueManager.hpp

```

00001 #ifndef __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP
00002 #define __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 //SEVMgr
00010 #include <sevmgr/SEVMGR_Types.hpp>
00011
00012 namespace SEVMGR {
00013
00014     extern const EventQueueID_T DEFAULT_EVENT_QUEUE_ID
00015 ;
00016
00017 }
00018 #endif // __SEVMGR_BAS_BASCONST_EVENTQUEUEMANAGER_HPP

```

23.19 sevmgr/basic/BasConst_SEVMGR_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace [SEVMGR](#)

23.20 BasConst_SEVMGR_Service.hpp

```

00001 #ifndef __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP
00002 #define __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 #include <string>
00008
00009 namespace SEVMGR {
00010
00011     // extern const std::string DEFAULT_SEVMGR_SERVICE_NAME;
00012
00013 }
00014
00015 #endif // __SEVMGR_BAS_BASCONST_SEVMGR_SERVICE_HPP

```

23.21 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

23.22 BasParserTypes.hpp

```

00001 #ifndef __SEVMGR_BAS_BASCOMPARSERTYPES_HPP
00002 #define __SEVMGR_BAS_BASCOMPARSERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL

```

```

00008 #include <string>
00009 // Boost
00010 // #define BOOST_SPIRIT_DEBUG
00011 #include <boost/spirit/home/classic/core.hpp>
00012 // #include <boost/spirit/home/classic/attribute.hpp>
00013 // #include <boost/spirit/home/classic/utility/functor_parser.hpp>
00014 #include <boost/spirit/home/classic/utility/loops.hpp>
00015 #include <boost/spirit/home/classic/utility/chset.hpp>
00016 #include <boost/spirit/home/classic/utility/confix.hpp>
00017 #include <boost/spirit/home/classic/iterator/file_iterator.hpp>
00018 // #include <boost/spirit/home/classic/actor/push_back_actor.hpp>
00019 // #include <boost/spirit/home/classic/actor/assign_actor.hpp>
00020
00021 namespace SEVMGR {
00022
00023 // //////////////////////////////////////
00024 //
00025 // Definition of Basic Types
00026 //
00027 // //////////////////////////////////////
00028 // For a file, the parsing unit is the character (char). For a string,
00029 // it is a "char const *".
00030 // typedef char const* iterator_t;
00031 typedef char char_t;
00032
00033 // The types of iterator, scanner and rule are then derived from
00034 // the parsing unit.
00035 typedef boost::spirit::classic::file_iterator<char_t> iterator_t;
00036 typedef boost::spirit::classic::scanner<iterator_t> scanner_t;
00037 typedef boost::spirit::classic::rule<scanner_t> rule_t;
00038
00039 // //////////////////////////////////////
00040 //
00041 // Parser related types
00042 //
00043 // //////////////////////////////////////
00044 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00045 ;
00046 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> uint2_p_t;
00047 ;
00048 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2>
00049 uint1_2_p_t;
00050
00051 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3>
00052 uint1_3_p_t;
00053
00054 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t;
00055 ;
00056 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
00057 uint1_4_p_t;
00058
00059 typedef boost::spirit::classic::chset<char_t> chset_t;
00060
00061 typedef boost::spirit::classic::impl::loop_traits<chset_t,
00062 unsigned int,
00063 unsigned int>::type repeat_p_t;
00064 ;
00065 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t;
00066 ;
00067 typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int>
00068 bounded1_2_p_t;
00069 typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int>
00070 bounded1_3_p_t;
00071 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t;
00072 ;
00073 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
00074 bounded1_4_p_t;
00075 }
00076 #endif // __SEVMGR_BAS_BASCOMPARSERTYPES_HPP

```

23.23 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

23.23.1 Function Documentation

23.23.1.1 `const stdair::Filename_T K_SEVMGR_DEFAULT_LOG_FILENAME` ("sevmgr_demo.log")

Default name and location for the log file.

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

23.23.1.2 `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\(\)](#).

23.23.1.3 `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\(\)](#).

23.23.2 Variable Documentation

23.23.2.1 `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\(\)](#).

23.24 sevmgr_demo.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <fstream>
00008 #include <vector>
00009 #include <list>
00010 #include <string>
00011 // Boost (Extended STL)
00012 // Boost Program Options
00013 #include <boost/program_options.hpp>
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/basic/ProgressStatusSet.hpp>
00017 #include <stdair/bom/EventStruct.hpp>
00018 #include <stdair/bom/BomDisplay.hpp>
00019 #include <stdair/service/Logger.hpp>
00020 #include <stdair/bom/BookingRequestStruct.hpp>
00021 #include <stdair/bom/BookingRequestTypes.hpp>
00022 #include <stdair/bom/EventStruct.hpp>
00023 // SEvMgr
00024 #include <sevmgr/SEVMGR_Service.hpp>
00025 #include <sevmgr/config/sevmgr-paths.hpp>
00026
00027 // Constants
00029 const stdair::Filename_T K_SEVMGR_DEFAULT_LOG_FILENAME
    ("sevmgr_demo.log");
00030
00032 const int K_SEVMGR_EARLY_RETURN_STATUS = 99;
00033
00034
00035 // Parsing of Options & Configuration
00037 int readConfiguration (int argc, char* argv[],
00038     stdair::Filename_T& ioLogFilename) {
00039
00040     // Declare a group of options that will be allowed only on command line
00041     boost::program_options::options_description generic ("Generic options");
00042     generic.add_options()
00043         ("prefix", "print installation prefix")
00044         ("version,v", "print version string")
00045         ("help,h", "produce help message");
00046
00047     // Declare a group of options that will be allowed both on command
00048     // line and in config file
00049     boost::program_options::options_description config ("Configuration");
00050     config.add_options()
00051         ("log,l",
00052             boost::program_options::value< std::string >(&ioLogFilename)->
00053             default_value(K_SEVMGR_DEFAULT_LOG_FILENAME),
00054             "Filepath for the logs")
00055         ;
00056
00057     // Hidden options, will be allowed both on command line and
00058     // in config file, but will not be shown to the user.
00059     boost::program_options::options_description hidden ("Hidden options");
00060     hidden.add_options()
00061         ("copyright",
00062             boost::program_options::value< std::vector<std::string> >(),
00063             "Show the copyright (license)");
00064
00065     boost::program_options::options_description cmdline_options;
00066     cmdline_options.add(generic).add(config).add(hidden);
00067
00068     boost::program_options::options_description config_file_options;
00069     config_file_options.add(config).add(hidden);
00070
00071     boost::program_options::options_description visible ("Allowed options");
00072     visible.add(generic).add(config);
00073
00074     boost::program_options::positional_options_description p;
00075     p.add ("copyright", -1);
00076
00077     boost::program_options::variables_map vm;
00078     boost::program_options::store (boost::program_options::command_line_parser (argc, argv).
00079         options (cmdline_options).positional(p).run(), vm);
00079

```



```

00080
00081     std::ifstream ifs ("sevMgr.cfg");
00082     boost::program_options::store (parse_config_file (ifs, config_file_options),
00083                                     vm);
00084     boost::program_options::notify (vm);
00085
00086     if (vm.count ("help")) {
00087         std::cout << visible << std::endl;
00088         return K_SEVMGR_EARLY_RETURN_STATUS;
00089     }
00090
00091     if (vm.count ("version")) {
00092         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
00093         << std::endl;
00094         return K_SEVMGR_EARLY_RETURN_STATUS;
00095     }
00096     if (vm.count ("prefix")) {
00097         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00098         return K_SEVMGR_EARLY_RETURN_STATUS;
00099     }
00100
00101     if (vm.count ("log")) {
00102         ioLogFilename = vm["log"].as< std::string >();
00103         std::cout << "Log filename is: " << ioLogFilename << std::endl;
00104     }
00105
00106     return 0;
00107 }
00108
00109
00110 // ////////////////////////////////// M A I N //////////////////////////////////
00111 int main (int argc, char* argv[]) {
00112
00113     // Output log File
00114     stdair::Filename_T lLogFilename;
00115
00116     // Call the command-line option parser
00117     const int lOptionParserStatus = readConfiguration (argc,
00118                                                         argv, lLogFilename);
00119
00120     if (lOptionParserStatus == K_SEVMGR_EARLY_RETURN_STATUS
00121 ) {
00122         return 0;
00123     }
00124
00125     // Set the log parameters
00126     std::ofstream logOutputFile;
00127     // Open and clean the log outputfile
00128     logOutputFile.open (lLogFilename.c_str());
00129     logOutputFile.clear();
00130
00131     // Set up the log parameters
00132     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00133
00134     SEVMGR::SEVMGR_Service sevMgrService (lLogParams);
00135
00136     // Build the default sample queue.
00137     STDAIR_LOG_DEBUG ("Build the default sample queue.");
00138     sevMgrService.buildSampleQueue();
00139
00140
00141     stdair::Count_T idx = 1;
00142     while (sevMgrService.isQueueDone() == false) {
00143
00144         // Pop the next event out of the event queue
00145         stdair::EventStruct lEventStruct;
00146         const stdair::ProgressStatusSet lPPS =
00147             sevMgrService.popEvent (lEventStruct);
00148
00149         // DEBUG
00150         STDAIR_LOG_DEBUG ("Poped event "<< idx << ": '"
00151                             << lEventStruct.describe() << "'.";
00152         STDAIR_LOG_DEBUG ("Progressss status: " << lPPS.describe());
00153
00154         // Iterate
00155         ++idx;
00156     }
00157
00158     // DEBUG
00159     STDAIR_LOG_DEBUG ("End of the simulation");
00160
00161     // Close the Log outputFile
00162     logOutputFile.close();
00163
00164     /*
00165     Note: as that program is not intended to be run on a server in
00166     production, it is better not to catch the exceptions. When it

```

```

00173     happens (that an exception is throwned), that way we get the
00174     call stack.
00175     */
00176
00177     return 0;
00178 }

```

23.25 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](#)

23.26 BomJSONExport.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
00011 #include <boost/regex.hpp>
00012 #endif // BOOST_VERSION >= 104100
00013 // StdAir
00014 #include <stdair/STDAIR_Service.hpp>
00015 #include <stdair/bom/EventStruct.hpp>
00016 // SEVMGR
00017 #include <sevmgr/bom/EventQueue.hpp>
00018 #include <sevmgr/bom/BomJSONExport.hpp>
00019
00020 #if BOOST_VERSION >= 104100
00021 namespace bpt = boost::property_tree;
00022 #else // BOOST_VERSION >= 104100
00023 namespace bpt {
00024     typedef char ptree;
00025 }
00026 #endif // BOOST_VERSION >= 104100
00027
00028 namespace SEVMGR {
00029
00030     // //////////////////////////////////////
00031     void BomJSONExport::
00032     jsonExportEventQueue (stdair::STDAIR_ServicePtr_T&
00033     ioSTDAIR_ServicePtr,
00034     std::ostream& oStream,
00035     const EventQueue& iEventQueue,
00036     const stdair::EventType::EN_EventType& iEventType) {
00037         // Retrieve the event list
00038         const stdair::EventList_T& lEventList = iEventQueue.getEventList
00039         ();
00040 #if BOOST_VERSION >= 104100
00041         // Create empty property tree objects

```

```

00042     bpt::ptree ptEvents;
00043     bpt::ptree pt;
00044
00045     // Browse the events
00046     for (stdair::EventList_T::const_iterator itEvent = lEventList.begin();
00047          itEvent != lEventList.end(); ++itEvent) {
00048         const stdair::EventStruct& lEvent = itEvent->second;
00049         const stdair::EventType::EN_EventType& lEventType =
00050             lEvent.getEventType();
00051
00052         const bool isEventTypeLastValue =
00053             (lEventType == stdair::EventType::LAST_VALUE);
00054         if (lEventType == iEventType || isEventTypeLastValue == true) {
00055
00056             // Delegate the JSON export to the dedicated service
00057             const std::string lCurrentEvent =
00058                 ioSTDAIR_ServicePtr->jsonExportEventObject (lEvent);
00059
00060             // Load the JSON formatted string into the property tree.
00061             // If reading fails (cannot open stream, parse error), an
00062             // exception is thrown.
00063             if (lCurrentEvent.empty () == false) {
00064                 bpt::ptree ptCurrentEvent;
00065                 std::istringstream lStrCurrentEvent(lCurrentEvent);
00066                 read_json (lStrCurrentEvent, ptCurrentEvent);
00067
00068                 // Put the current inventory tree in the events array
00069                 ptEvents.push_back(std::make_pair("", ptCurrentEvent));
00070             }
00071         }
00072     }
00073
00074     // Store the events array tree into the global tree
00075     pt.add_child ("events", ptEvents);
00076
00077     // Write the property tree into the JSON stream.
00078     write_json (oStream, pt);
00079
00080 #endif // BOOST_VERSION >= 104100
00081 }
00082
00083 }

```

23.27 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](#)

23.28 BomJSONExport.hpp

```

00001 #ifndef __SEVMGR_BOM_BOMJSONEXPORT_HPP
00002 #define __SEVMGR_BOM_BOMJSONEXPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // Boost Property Tree
00010 #if BOOST_VERSION >= 104100

```

```

00011 #include <boost/property_tree/ptree.hpp>
00012 #include <boost/property_tree/json_parser.hpp>
00013 #endif // BOOST_VERSION >= 104100
00014 // StdAir
00015 #include <stdair/stdair_service_types.hpp>
00016 #include <stdair/bom/EventTypes.hpp>
00017
00018 #if BOOST_VERSION >= 104100
00019     namespace bpt = boost::property_tree;
00020 #else // BOOST_VERSION >= 104100
00021     namespace bpt {
00022         typedef char ptree;
00023     }
00024 #endif // BOOST_VERSION >= 104100
00025
00026 namespace SEVMGR {
00027
00028     class EventQueue;
00029
00030     class BomJSONExport {
00031     public:
00032         // ////////////////////////////////// Export support methods //////////////////////////////////
00033
00034         static void jsonExportEventQueue (
00035             stdair::STDAIR_ServicePtr_T&,
00036             std::ostream&,
00037             const EventQueue&,
00038             const stdair::EventType::EN_EventType&);
00039     };
00040 }
00041 #endif // __SEVMGR_BOM_BOMJSONEXPORT_HPP

```

23.29 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](#)

23.30 EventQueue.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_exceptions.hpp>
00008 #include <stdair/basic/BasConst_Event.hpp>
00009 #include <stdair/bom/EventStruct.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // SEvMgr
00012 #include <sevmgr/basic/BasConst_EventQueueManager.hpp>
00013
00014 #include <sevmgr/bom/EventQueue.hpp>
00015
00016 namespace SEVMGR {
00017
00018     // //////////////////////////////////////
00019     EventQueue::EventQueue()
00020         : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
00021           _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00022                           stdair::DEFAULT_PROGRESS_STATUS) {
00023     }

```

```

00023
00024 // //////////////////////////////////////
00025 EventQueue::EventQueue (const Key_T& iKey)
00026 : _key (iKey), _parent (NULL),
00027   _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00028                   stdair::DEFAULT_PROGRESS_STATUS) {
00029 }
00030
00031 // //////////////////////////////////////
00032 EventQueue::EventQueue (const EventQueue& iEventQueue)
00033 : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
00034   _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00035                   stdair::DEFAULT_PROGRESS_STATUS) {
00036     assert (false);
00037 }
00038
00039 // //////////////////////////////////////
00040 EventQueue::~EventQueue () {
00041   _eventList.clear();
00042 }
00043
00044 // //////////////////////////////////////
00045 std::string EventQueue::toString() const {
00046   std::ostringstream oStr;
00047   oStr << "(" << _eventList.size() << " "
00048         << _progressStatus.getCurrentNb() << "/" << "
00049         << _progressStatus.getExpectedNb() << " "
00050         << _progressStatus.getActualNb() << ")";
00051   return oStr.str();
00052 }
00053
00054 // //////////////////////////////////////
00055 std::string EventQueue::display() const {
00056   std::ostringstream oStr;
00057
00058   oStr << toString();
00059
00060   return oStr.str();
00061 }
00062
00063 // //////////////////////////////////////
00064 std::string EventQueue::list () const {
00065   std::ostringstream oStr;
00066   oStr << describeKey () << std::endl;
00067   oStr << toString() << std::endl;
00068
00069   // Browse the events
00070   for (stdair::EventList_T::const_iterator itEvent = _eventList.
begin();
00071        itEvent != _eventList.end(); ++itEvent) {
00072     const stdair::EventStruct& lEvent = itEvent->second;
00073
00074     oStr << lEvent.describe();
00075   }
00076
00077   return oStr.str();
00078 }
00079
00080 // //////////////////////////////////////
00081 std::string EventQueue::
00082 list (const stdair::EventType::EN_EventType& iType) const {
00083   std::ostringstream oStr;
00084   oStr << describeKey () << std::endl;
00085   oStr << toString() << std::endl;
00086   oStr << "List " << stdair::EventType::getLabel(iType)
00087         << " events:" << std::endl;
00088
00089   // Browse the events
00090   for (stdair::EventList_T::const_iterator itEvent = _eventList.
begin();
00091        itEvent != _eventList.end(); ++itEvent) {
00092     const stdair::EventStruct& lEvent = itEvent->second;
00093
00094     if (lEvent.getEventType() == iType) {
00095       oStr << lEvent.describe();
00096     }
00097   }
00098   return oStr.str();
00099 }
00100
00101 // //////////////////////////////////////
00102 stdair::Count_T EventQueue::getQueueSize () const {
00103   return _eventList.size();
00104 }
00105
00106 // //////////////////////////////////////
00107 bool EventQueue::isEmpty () const {

```

```

00108     return _eventList.empty();
00109 }
00110
00111 // //////////////////////////////////////
00112 bool EventQueue::isQueueDone () const {
00113     const bool isQueueEmpty = _eventList.empty();
00114     return isQueueEmpty;
00115 }
00116
00117 // //////////////////////////////////////
00118 void EventQueue::reset () {
00119     // Reset only the current number of events, not the expected one
00120     _progressStatus.reset();
00121
00122     // Empty the list of events
00123     _eventList.clear();
00124
00125     // Reset the progress statuses for all the event types
00126     for (ProgressStatusMap_T::iterator itProgressStatus =
00127         _progressStatusMap.begin();
00128         itProgressStatus != _progressStatusMap.end(); ++
00129         itProgressStatus) {
00130         stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00131         lProgressStatus.reset();
00132     }
00133
00134 // //////////////////////////////////////
00135 bool EventQueue::
00136 hasProgressStatus (const stdair::EventType::EN_EventType&
00137 iType) const {
00138     bool hasProgressStatus = true;
00139
00140     // Retrieve the ProgressStatus structure corresponding to the
00141     // given event type
00142     ProgressStatusMap_T::const_iterator itProgressStatus =
00143         _progressStatusMap.find (iType);
00144     if (itProgressStatus == _progressStatusMap.end()) {
00145         //
00146         STDAIR_LOG_DEBUG ("No ProgressStatus structure can be retrieved in the "
00147             << "EventQueue: " << display());
00148
00149         hasProgressStatus = false;
00150     }
00151
00152     return hasProgressStatus;
00153 }
00154
00155 // //////////////////////////////////////
00156 const stdair::Count_T& EventQueue::
00157 getCurrentNbOfEvents (const
00158 stdair::EventType::EN_EventType& iType) const {
00159     // Retrieve the ProgressStatus structure corresponding to the
00160     // given event type
00161     ProgressStatusMap_T::const_iterator itProgressStatus =
00162         _progressStatusMap.find (iType);
00163     if (itProgressStatus == _progressStatusMap.end()) {
00164         //
00165         STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00166             << "EventQueue: " << display());
00167         assert (false);
00168     }
00169
00170     const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00171     return lProgressStatus.getCurrentNb();
00172 }
00173
00174 // //////////////////////////////////////
00175 const stdair::Count_T& EventQueue::
00176 getExpectedTotalNbOfEvents (const
00177 stdair::EventType::EN_EventType& iType) const {
00178     // Retrieve the ProgressStatus structure corresponding to the
00179     // given event type
00180     ProgressStatusMap_T::const_iterator itProgressStatus =
00181         _progressStatusMap.find (iType);
00182     if (itProgressStatus == _progressStatusMap.end()) {
00183         std::ostringstream oStr;
00184         oStr << "No ProgressStatus structure can be retrieved in the EventQueue '
00185             << display() << "'. The EventQueue should be initialised,
00186             e.g., by "
00187             << "calling a buildSampleBom() method.";
00188         STDAIR_LOG_ERROR (oStr.str());

```

```

00189         throw EventQueueException (oStr.str());
00190     }
00191
00192     const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00193     return lProgressStatus.getExpectedNb();
00194 }
00195
00196 // //////////////////////////////////////
00197 const stdair::Count_T& EventQueue::
00198 getActualTotalNbOfEvents (const
stdair::EventType::EN_EventType& iType) const {
00199
00200     // Retrieve the ProgressStatus structure corresponding to the
00201     // given event type
00202     ProgressStatusMap_T::const_iterator itProgressStatus =
00203     _progressStatusMap.find (iType);
00204     if (itProgressStatus == _progressStatusMap.end()) {
00205         //
00206         STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00207             << "EventQueue: " << display());
00208         assert (false);
00209     }
00210
00211     const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00212     return lProgressStatus.getActualNb();
00213 }
00214
00215 // //////////////////////////////////////
00216 void EventQueue::updateStatus (const
stdair::EventType::EN_EventType& iType,
00217     const stdair::ProgressStatus& iProgressStatus)
00218 {
00219     // Retrieve, if existing, the ProgressStatus structure
00220     // corresponding to the given event type
00221     ProgressStatusMap_T::iterator itProgressStatus =
00222     _progressStatusMap.find (iType);
00223     if (itProgressStatus == _progressStatusMap.end()) {
00224         const bool hasInsertBeenSuccessful =
00225         _progressStatusMap.insert (ProgressStatusMap_T
::
00226             value_type (iType, iProgressStatus)).second;
00227
00228         if (hasInsertBeenSuccessful == false) {
00229             STDAIR_LOG_ERROR ("No progress_status can be inserted "
00230                 << "for the following event type: "
00231                 << stdair::EventType::getLabel(iType)
00232                 << ". EventQueue: " << toString());
00233             throw stdair::EventException ("No progress_status can be inserted for
the "
00234                 "following event type: "
00235                 + stdair::EventType::getLabel(iType)
00236                 + ". EventQueue: " + toString());
00237         }
00238
00239         return;
00240     }
00241
00242     stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00243
00244     // Update the progress status
00245     const stdair::Count_T& lCurrentNb = iProgressStatus.getCurrentNb();
00246     lProgressStatus.setCurrentNb (lCurrentNb);
00247
00248     const stdair::Count_T& lExpectedNb = iProgressStatus.getExpectedNb();
00249     lProgressStatus.setExpectedNb(lProgressStatus.getExpectedNb() + lExpectedNb
);
00250
00251     const stdair::Count_T& lActualNb = iProgressStatus.getActualNb();
00252     lProgressStatus.setActualNb (lProgressStatus.getActualNb() + lActualNb);
00253 }
00254
00255 // //////////////////////////////////////
00256 void EventQueue::
00257 addStatus (const stdair::EventType::EN_EventType& iType,
00258     const stdair::NbOfEvents_T& iExpectedTotalNbOfEvents) {
00259
00260     // Initialise the progress status object
00261     const stdair::Count_T lExpectedTotalNbOfEventsInt =
00262     static_cast<const stdair::Count_T> (std::floor (iExpectedTotalNbOfEvents)
);
00263     const stdair::ProgressStatus lProgressStatus (lExpectedTotalNbOfEventsInt);
00264
00265     // Update the progress status for the given event type
00266     updateStatus (iType, lProgressStatus);
00267
00268     // Update the overall progress status

```

```

00269     const stdair::Count_T lExpectedNb =
00270         static_cast<const stdair::Count_T> (_progressStatus.
getExpectedNb()
00271             + iExpectedTotalNbOfEvents);
00272     _progressStatus.setExpectedNb (lExpectedNb);
00273
00274     const stdair::Count_T lActualNb =
00275         static_cast<const stdair::Count_T> (_progressStatus.
getActualNb()
00276             + iExpectedTotalNbOfEvents);
00277     _progressStatus.setActualNb (lActualNb);
00278
00279 }
00280
00281 // //////////////////////////////////////
00282 void EventQueue::updateStatus (const
stdair::EventType::EN_EventType& iType,
00283     const stdair::NbOfEvents_T& iActualNbOfEvents)
{
00284
00285     // Initialise the progress status object for the type key
00286     stdair::Count_T lActualNbOfEventsInt =
00287         static_cast<const stdair::Count_T> (std::floor (iActualNbOfEvents));
00288
00289     // Update the progress status for the corresponding content type key
00290     ProgressStatusMap_T::iterator itProgressStatus =
00291         _progressStatusMap.find (iType);
00292     if (itProgressStatus != _progressStatusMap.end()) {
00293
00294         //
00295         stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00296
00297         // Update the overall progress status
00298         const stdair::Count_T lActualEventTypeNb = lProgressStatus.getActualNb();
00299         const stdair::Count_T lActualTotalNb = _progressStatus.
getActualNb();
00300         _progressStatus.setActualNb (lActualTotalNb +
iActualNbOfEvents - lActualEventTypeNb);
00301
00302         // Update the progress status for the corresponding type key
00303         lProgressStatus.setActualNb (lActualNbOfEventsInt);
00304     }
00305 }
00306
00307 // //////////////////////////////////////
00308 void EventQueue::setStatus (const
stdair::EventType::EN_EventType& iType,
00309     const stdair::ProgressStatus& iProgressStatus) {
00310
00311     // Retrieve the ProgressStatus structure corresponding to the
00312     // given event type
00313     ProgressStatusMap_T::iterator itProgressStatus =
00314         _progressStatusMap.find (iType);
00315     // assert (itProgressStatus != _progressStatusMap.end());
00316     if (itProgressStatus != _progressStatusMap.end()) {
00317         // Update the ProgressStatus structure
00318         itProgressStatus->second = iProgressStatus;
00319     }
00320 }
00321
00322 // //////////////////////////////////////
00323 const stdair::ProgressStatus& EventQueue::
00324 getStatus (const stdair::EventType::EN_EventType& iType) const {
00325
00326     // Retrieve the ProgressStatus structure corresponding to the
00327     // given event type
00328     ProgressStatusMap_T::const_iterator itProgressStatus =
00329         _progressStatusMap.find (iType);
00330     if (itProgressStatus == _progressStatusMap.end()) {
00331         std::ostream oStr;
00332         oStr << "No ProgressStatus structure can be retrieved in the EventQueue "
00333
00334         << display() << "' for the following event type: "
00335         << stdair::EventType::getLabel(iType) << ".";
00336         //
00337         STDAIR_LOG_ERROR (oStr.str());
00338         throw EventQueueException (oStr.str());
00339     }
00340     assert(itProgressStatus != _progressStatusMap.end());
00341
00342     const stdair::ProgressStatus& oProgressStatus = itProgressStatus->second;
00343     return oProgressStatus;
00344 }
00345
00346 // //////////////////////////////////////
00347 stdair::ProgressPercentage_T EventQueue::
calculateProgress (const stdair::EventType::EN_EventType&

```



```

iType) const {
00348
00349     // Retrieve the ProgressStatus structure corresponding to the
00350     // given event type
00351     ProgressStatusMap_T::const_iterator itProgressStatus =
00352         _progressStatusMap.find (iType);
00353     if (itProgressStatus == _progressStatusMap.end()) {
00354         //
00355         STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00356             << "EventQueue: " << display());
00357         assert (false);
00358     }
00359     const stdair::ProgressStatus& lProgressStatus = itProgressStatus->second;
00360     return lProgressStatus.progress();
00361 }
00362
00363 // //////////////////////////////////////
00364 stdair::ProgressStatusSet EventQueue::popEvent (
00365     stdair::EventStruct& ioEventStruct) {
00366
00367     if (_eventList.empty() == true) {
00368         std::ostream oStr;
00369         oStr << "The event queue '" << describeKey() << "' is empty. "
00370             << "No event can be popped.";
00371         //
00372         STDAIR_LOG_ERROR (oStr.str());
00373         throw EventQueueException (oStr.str());
00374     }
00375
00376     // Get an iterator on the first event (sorted by date-time stamps)
00377     stdair::EventList_T::iterator itEvent = _eventList.begin();
00378
00379     ioEventStruct = itEvent->second;
00380     // Retrieve the event type
00381     const stdair::EventType& lEventType = ioEventStruct.
00382         getEventType();
00383     stdair::ProgressStatusSet oProgressStatusSet (lEventType);
00384
00385     // Update the (current number part of the) overall progress status,
00386     // to account for the event that is being popped out of the event
00387     // queue.
00388     ++_progressStatus;
00389
00390     // Remove the event, which has just been retrieved
00391     _eventList.erase (itEvent);
00392
00393     // Retrieve the progress status specific to that event type
00394     stdair::ProgressStatus lEventTypeProgressStatus = getStatus (
00395         lEventType);
00396
00397     // Increase the current number of events
00398     ++lEventTypeProgressStatus;
00399
00400     // Store back the progress status
00401     setStatus (lEventType, lEventTypeProgressStatus);
00402
00403     // Update the progress status of the progress status set, specific to
00404     // the event type.
00405     oProgressStatusSet.setTypeSpecificStatus (lEventTypeProgressStatus);
00406
00407     // Update the overall progress status of the progress status set.
00408     oProgressStatusSet.setOverallStatus (_progressStatus);
00409
00410     //
00411     return oProgressStatusSet;
00412 }
00413
00414 // //////////////////////////////////////
00415 bool EventQueue::addEvent (stdair::EventStruct&
00416     ioEventStruct) {
00417     bool insertionSucceeded =
00418         _eventList.insert (stdair::EventListElement_T (ioEventStruct.
00419             getEventTimeStamp(),
00420                 ioEventStruct)).second;
00421
00422     const unsigned int idx = 0;
00423     while (insertionSucceeded == false && idx != 1e3) {
00424         // Increment the date-time stamp (expressed in milliseconds)
00425         ioEventStruct.incrementEventTimeStamp();
00426
00427         // Retry to insert into the event queue
00428         insertionSucceeded =
00429             _eventList.insert (stdair::EventListElement_T (ioEventStruct.
00430                 getEventTimeStamp(),
00431                     ioEventStruct)).second;
00432     }
00433 }

```

```

00459     }
00460     assert (idx != 1e3);
00461
00462     return insertionSucceeded;
00463 }
00464
00465 // //////////////////////////////////////
00466 bool EventQueue::hasEventDateTime (const
stdair::DateTime_T& iDateTime) {
00467
00468     bool hasSearchEventBeenSuccessful = true;
00469
00470     const stdair::Duration_T lDuration =
00471         iDateTime - stdair::DEFAULT_EVENT_OLDEST_DATETIME;
00472     const stdair::LongDuration_T lDateTimeStamp =
00473         lDuration.total_milliseconds();
00474
00475     // Searches the container for an element with iDateTime as key
00476     stdair::EventList_T::iterator itEvent =
00477         _eventList.find (lDateTimeStamp);
00478
00479     // An iterator to map::end means the specified key has not found in the
00480     // container.
00481     if (itEvent == _eventList.end()) {
00482         hasSearchEventBeenSuccessful = false;
00483     }
00484
00485     return hasSearchEventBeenSuccessful;
00486 }
00487
00488 }
00489
00490 }
00491
00492 }
00493
00494 }

```

23.31 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](#)

23.32 EventQueue.hpp

```

00001 #ifndef __SEVMGR_BOM_EVENTQUEUE_HPP
00002 #define __SEVMGR_BOM_EVENTQUEUE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////

```

```

00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/basic/ProgressStatusSet.hpp>
00014 #include <stdair/basic/EventType.hpp>
00015 #include <stdair/bom/BomAbstract.hpp>
00016 #include <stdair/bom/EventTypes.hpp>
00017 // SEvMgr
00018 #include <sevmgr/bom/EventQueueKey.hpp>
00019 #include <sevmgr/bom/EventQueueTypes.hpp>
00020 #include <sevmgr/SEVMGR_Types.hpp>
00021
00023 namespace stdair {
00024     class FacBomManager;
00025     template <typename BOM> class FacBom;
00026 }
00027
00028 namespace SEVMGR {
00029
00068     class EventQueue : public stdair::BomAbstract {
00069     public:
00070         template <typename BOM> friend class stdair::FacBom;
00071         friend class stdair::FacBomManager;
00072
00073         // ////////// Type definitions //////////
00077         typedef EventQueueKey Key_T;
00078
00079     public:
00081         // ////////// Getters //////////
00083         const Key_T& getKey () const {
00084             return _key;
00085         }
00086
00088         BomAbstract* const getParent () const {
00089             return _parent;
00090         }
00091
00093         const stdair::EventList_T& getEventList () const {
00094             return _eventList;
00095         }
00096
00098         const stdair::HolderMap_T& getHolderMap () const {
00099             return _holderMap;
00100         }
00101
00103         const stdair::ProgressStatus& getStatus () const {
00104             return _progressStatus;
00105         }
00107         const stdair::Count_T& getCurrentNbOfEvents () const {
00108             return _progressStatus.getCurrentNb();
00109         }
00111         const stdair::Count_T& getExpectedTotalNbOfEvents
00112         () const {
00113             return _progressStatus.getExpectedNb();
00114         }
00115         const stdair::Count_T& getActualTotalNbOfEvents ()
00116         const {
00117             return _progressStatus.getActualNb();
00118         }
00123         const stdair::ProgressStatus& getStatus (const
00124         stdair::EventType::EN_EventType&) const;
00126         const stdair::Count_T& getCurrentNbOfEvents (const
00127         stdair::EventType::EN_EventType&) const;
00129         const stdair::Count_T& getExpectedTotalNbOfEvents
00130         (const stdair::EventType::EN_EventType&) const;
00132         const stdair::Count_T& getActualTotalNbOfEvents (
00133         const stdair::EventType::EN_EventType&) const;
00136         bool hasProgressStatus (const
00137         stdair::EventType::EN_EventType&) const;
00138     public:
00139         // ////////// Setters //////////
00141         void setStatus (const stdair::ProgressStatus& iProgressStatus) {
00142             _progressStatus = iProgressStatus;
00143         }
00145         void setStatus (const stdair::Count_T& iCurrentNbOfEvents,
00146             const stdair::Count_T& iExpectedTotalNbOfEvents,
00147             const stdair::Count_T& iActualTotalNbOfEvents) {

```

```

00148     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00149     _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
00150     _progressStatus.setActualNb (iActualTotalNbOfEvents);
00151 }
00153 void setStatus (const stdair::Count_T& iCurrentNbOfEvents,
00154                const stdair::Count_T& iActualTotalNbOfEvents) {
00155     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00156     _progressStatus.setActualNb (iActualTotalNbOfEvents);
00157 }
00159 void setCurrentNbOfEvents (const stdair::Count_T&
00160 iCurrentNbOfEvents) {
00161     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00162 }
00163 void setExpectedTotalNbOfEvents (const
00164 stdair::Count_T& iExpectedTotalNbOfEvents) {
00165     _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
00166 }
00171 void setStatus (const stdair::EventType::EN_EventType& iType,
00172                const stdair::ProgressStatus& iProgressStatus);
00173
00174
00175 public:
00176     // //////////// Display support methods ////////////
00182     void toStream (std::ostream& ioOut) const {
00183         ioOut << toString();
00184     }
00185
00191     void fromStream (std::istream& ioIn) {
00192     }
00193
00197     std::string toString () const;
00198
00202     std::string list () const;
00203
00208     std::string list (const stdair::EventType::EN_EventType&) const;
00209
00213     const std::string describeKey () const {
00214         return _key.toString();
00215     }
00216
00217     /*
00218     * Display the full content of the event queue, with all its
00219     * event structure.
00220     *
00221     * That method can be very consuming (in time, CPU and memory)
00222     * when there are a lot of event structures (e.g., several hundreds
00223     * of thousands). Call it only for debug purposes.
00224     */
00225     std::string display () const;
00226
00227
00228 public:
00229     // //////////// Business methods ////////////
00234     void reset ();
00235
00249     stdair::ProgressStatusSet popEvent (stdair::EventStruct&);
00250
00271     bool addEvent (stdair::EventStruct&);
00272
00276     bool hasEventDateTime (const stdair::DateTime_T&);
00277
00283     bool isQueueDone () const;
00284
00298     void addStatus (const stdair::EventType::EN_EventType&,
00299                    const stdair::NbOfRequests_T& iExpectedTotalNbOfEvents);
00300
00309     void updateStatus (const stdair::EventType::EN_EventType&,
00310                       const stdair::ProgressStatus& iProgressStatus);
00311
00325     void updateStatus (const stdair::EventType::EN_EventType&,
00326                       const stdair::NbOfEvents_T& iActualTotalNbOfEvents);
00327
00338     stdair::ProgressPercentage_T calculateProgress () const {
00339         return _progressStatus.progress();
00340     }
00341
00352     stdair::ProgressPercentage_T calculateProgress (const
00353 stdair::EventType::EN_EventType&) const;
00354
00355 public:
00356     // //////////// Debug methods ////////////
00358     stdair::Count_T getQueueSize () const;
00359
00361     bool isQueueEmpty () const;
00362

```

```

00363
00364 protected:
00365     // ////////// Constructors and destructors //////////
00367     EventQueue (const Key_T&);
00369     EventQueue (const EventQueue&);
00371     ~EventQueue ();
00372 private:
00374     EventQueue ();
00375
00376
00377 protected:
00378     // ////////// Attributes //////////
00382     Key_T _key;
00383
00387     BomAbstract* _parent;
00388
00394     stdair::HolderMap_T _holderMap;
00395
00399     stdair::EventList_T _eventList;
00400
00404     stdair::ProgressStatus _progressStatus;
00405
00411     ProgressStatusMap_T _progressStatusMap
00412 ;
00413 };
00414 }
00415 #endif // __SEVMGR_BOM_EVENTQUEUE_HPP

```

23.33 sevmgr/bom/EventQueueKey.cpp File Reference

```

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

```

Namespaces

- namespace [SEVMGR](#)

23.34 EventQueueKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // SEvMgr
00007 #include <sevmgr/bom/EventQueueKey.hpp>
00008
00009 namespace SEVMGR {
00010
00011     // //////////////////////////////////////
00012     EventQueueKey::EventQueueKey (const EventQueueID_T&
00013     iEventQueueID)
00014     : _eventQueueID (iEventQueueID) {
00015     }
00016     // //////////////////////////////////////
00017     EventQueueKey::EventQueueKey (const EventQueueKey& iKey)
00018     : _eventQueueID (iKey._eventQueueID) {
00019     }
00020
00021     // //////////////////////////////////////
00022     EventQueueKey::~EventQueueKey () {
00023     }
00024
00025     // //////////////////////////////////////
00026     void EventQueueKey::toStream (std::ostream& ioOut)
00027     const {
00028         ioOut << "EventQueueKey: " << toString() << std::endl;
00029     }
00030     // //////////////////////////////////////
00031     void EventQueueKey::fromStream (std::istream& ioIn)
00032     {
00033     }
00034     // //////////////////////////////////////

```

```

00034     const std::string EventQueueKey::toString() const {
00035         std::ostringstream ostr;
00036         ostr << _eventQueueID;
00037         return ostr.str();
00038     }
00039
00040 }
```

23.35 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](#)

23.36 EventQueueKey.hpp

```

00001 #ifndef __SEVMGR_BOM_EVENTQUEUEKEY_HPP
00002 #define __SEVMGR_BOM_EVENTQUEUEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_event_types.hpp>
00010 #include <stdair/bom/KeyAbstract.hpp>
00011 //SEVMgr
00012 #include <sevmgr/SEVMGR_Types.hpp>
00013
00014 namespace SEVMGR {
00015
00016     struct EventQueueKey : public stdair::KeyAbstract {
00017     private:
00020         // /////////// Default constructor ///////////
00021         EventQueueKey () { };
00022     public:
00024         // /////////// Construction ///////////
00026         EventQueueKey (const EventQueueID_T&);
00027         EventQueueKey (const EventQueueKey&);
00029         ~EventQueueKey ();
00030
00031         // /////////// Getters ///////////
00033         const EventQueueID_T& getEventQueueID() const
00034     {
00035         return _eventQueueID;
00036     }
00037     // /////////// Display support methods ///////////
00040     void toStream (std::ostream& ioOut) const;
00041
00044     void fromStream (std::istream& ioIn);
00045
00051     const std::string toString() const;
00052
00053     private:
00055         // /////////// Attributes ///////////
00057         EventQueueID_T _eventQueueID;
00058     };
00059
00060 }
```

```
00061 #endif // __SEVMGR_BOM_EVENTQUEUEKEY_HPP
```

23.37 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](#)

23.38 EventQueueTypes.hpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __SEVMGR_BOM_EVENTQUEUEUETYPES_HPP
00003 #define __SEVMGR_BOM_EVENTQUEUEUETYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace SEVMGR {
00015
00016     // Forward declarations.
00017     class EventQueue;
00018
00020     typedef std::list<EventQueue*> EventQueueList_T;
00021
00023     typedef std::map<const stdair::MapKey_T, EventQueue*> EventQueueMap_T
00024 ;
00025 }
00026 #endif // __SEVMGR_BOM_EVENTQUEUEUETYPES_HPP
```

23.39 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](#)

23.40 EventQueueManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/make_shared.hpp>
00008 // StdAir
00009 #include <stdair/basic/ProgressStatusSet.hpp>
00010 #include <stdair/basic/EventType.hpp>
00011 #include <stdair/basic/BasConst_Event.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/EventStruct.hpp>
00014 #include <stdair/bom/BookingRequestStruct.hpp>
00015 #include <stdair/bom/BreakPointStruct.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 #include <stdair/STDAIR_Service.hpp>
00018 // SEvMgr
00019 #include <sevmgr/bom/EventQueue.hpp>
00020 #include <sevmgr/command/EventQueueManager.hpp>
00021 >
00022 namespace SEVMGR {
00023
00024 // //////////////////////////////////////
00025 void EventQueueManager::
00026 buildSampleQueue (stdair::STDAIR_ServicePtr_T lSTDAIR_ServicePtr,
00027                  EventQueue& ioEventQueue) {
00028
00029     // Total number of booking requests into the queue
00030     stdair::Count_T lNbOfBookingRequests (2);
00031     addStatus(ioEventQueue, stdair::EventType::BKG_REQ, lNbOfBookingRequests);
00032
00033     // Create a shared pointer on a first booking request
00034     // Date of the request (15-MAY-2011)
00035     const stdair::BookingRequestStruct& lBookingRequest =
00036         buildSampleBookingRequest (lSTDAIR_ServicePtr);
00037     const stdair::BookingRequestPtr_T lBookingRequest_ptr =
00038         boost::make_shared<stdair::BookingRequestStruct> (lBookingRequest);
00039
00040     // Create an event structure
00041     stdair::EventStruct lEventStruct (stdair::EventType::BKG_REQ,
00042                                     lBookingRequest_ptr);
00043
00044     // Add the event into the queue
00045     addEvent(ioEventQueue, lEventStruct);
00046
00047     // Create a second shared pointer on a second booking request
00048     // Date of the request (22-JAN-2010)
00049     const bool isForCRS = true;
00050     const stdair::BookingRequestStruct& lBookingRequestForCRS =
00051         buildSampleBookingRequest (lSTDAIR_ServicePtr, isForCRS);
00052     const stdair::BookingRequestPtr_T lBookingRequestForCRS_ptr =
00053         boost::make_shared<stdair::BookingRequestStruct> (lBookingRequestForCRS);
00054
00055     // Create an event structure
00056     stdair::EventStruct lEventStructForCRS (stdair::EventType::BKG_REQ,
00057                                             lBookingRequestForCRS_ptr);
00058
00059     // Add the event into the queue
00060     addEvent(ioEventQueue, lEventStructForCRS);
00061
00062     // Total number of break points into the queue
00063     stdair::Count_T lNbOfBreakPoints (2);
00064     addStatus(ioEventQueue, stdair::EventType::BRK_PT, lNbOfBreakPoints);
00065
00066     // Create a shared pointer on a second break point
00067     // Date of the break point (21-JAN-2010)
00068     const stdair::Date_T lBP1Date (2010, boost::gregorian::Jan, 21);
00069     // Time of the break point (00:00)
00070     const stdair::Duration_T lBP1Time (0, 0, 0);
00071     // Date-time of the break point (made of the date and time above)
00072     const stdair::DateTime_T lBP1DateTime (lBP1Date, lBP1Time);
00073     const stdair::BreakPointPtr_T lBreakPoint1_ptr =
00074         boost::make_shared<stdair::BreakPointStruct> (lBP1DateTime);
00075
00076     // Create an event structure

```



```

00077     stdair::EventStruct lEventBreakPoint1 (stdair::EventType::BRK_PT,
00078                                           lBreakPoint1_ptr);
00079
00080     // Add the event into the queue
00081     addEvent(ioEventQueue, lEventBreakPoint1);
00082
00083     // Create a shared pointer on a second break point
00084     // Date of the break point (14-MAY-2011)
00085     const stdair::Date_T lBP2Date (2011, boost::gregorian::May, 14);
00086     // Time of the break point (00:00)
00087     const stdair::Duration_T lBP2Time (0, 0, 0);
00088     // Date-time of the break point (made of the date and time above)
00089     const stdair::DateTime_T lBP2DateTime (lBP2Date, lBP2Time);
00090
00091     // TODO: understand why the form above does not work.
00092     const stdair::BreakPointPtr_T lBreakPoint2_ptr =
00093         boost::make_shared<stdair::BreakPointStruct> (lBP2DateTime);
00094
00095     // Create an event structure
00096     stdair::EventStruct lEventBreakPoint2 (stdair::EventType::BRK_PT,
00097                                           lBreakPoint2_ptr);
00098
00099     // Add the event into the queue
00100     addEvent(ioEventQueue, lEventBreakPoint2);
00101
00102 }
00103
00104 // //////////////////////////////////////
00105 stdair::BookingRequestStruct EventQueueManager::
00106 buildSampleBookingRequest (stdair::STDAIR_ServicePtr_T lSTDAIR_ServicePtr,
00107                           const bool isForCRS) {
00108
00109     // Delegate the booking request building to the dedicated service
00110     stdair::BookingRequestStruct oBookingRequest =
00111         lSTDAIR_ServicePtr->buildSampleBookingRequest (isForCRS);
00112
00113     return oBookingRequest;
00114 }
00115
00116 // //////////////////////////////////////
00117 void EventQueueManager::reset (EventQueue& ioEventQueue) {
00118
00119     ioEventQueue.reset();
00120 }
00121
00122 // //////////////////////////////////////
00123 bool EventQueueManager::
00124 hasProgressStatus (const EventQueue& iEventQueue,
00125                   const stdair::EventType::EN_EventType& iEventType) {
00126
00127     const bool hasProgressStatus = iEventQueue.hasProgressStatus(iEventType);
00128
00129     //
00130     return hasProgressStatus;
00131 }
00132
00133 // //////////////////////////////////////
00134 void EventQueueManager::addEvent (EventQueue& ioEventQueue,
00135                                   stdair::EventStruct& iEventStruct) {
00136
00137     ioEventQueue.addEvent(iEventStruct);
00138 }
00139
00140 // //////////////////////////////////////
00141 const std::string EventQueueManager::
00142 list (const EventQueue& iEventQueue) {
00143
00144     const std::string& lEventListStr = iEventQueue.list();
00145
00146     //
00147     return lEventListStr;
00148 }
00149
00150 // //////////////////////////////////////
00151 const std::string EventQueueManager::
00152 list (const EventQueue& iEventQueue,
00153       const stdair::EventType::EN_EventType& iEventType) {
00154
00155     const std::string& lEventListStr =
00156         iEventQueue.list(iEventType);
00157
00158     //
00159     return lEventListStr;
00160 }
00161
00162 // //////////////////////////////////////
00163 const std::string EventQueueManager::
00164 list (const EventQueue& iEventQueue,
00165       const stdair::EventType::EN_EventType& iEventType) {
00166
00167     const std::string& lEventListStr =
00168         iEventQueue.list(iEventType);
00169
00170     //
00171     return lEventListStr;
00172 }
00173
00174 // //////////////////////////////////////

```

```

00179     const std::string EventQueueManager::
00180     describeKey (const EventQueue& iEventQueue) {
00181
00182         const std::string& lEventQueueKeyStr = iEventQueue.describeKey();
00183         //
00184         return lEventQueueKeyStr;
00185     }
00186
00187     //
00188     // //////////////////////////////////////
00189     stdair::ProgressStatusSet EventQueueManager::
00190     popEvent (EventQueue& ioEventQueue,
00191             stdair::EventStruct& iEventStruct) {
00192
00193         try {
00194             const stdair::ProgressStatusSet lProgressStatusSet
00195             = ioEventQueue.popEvent (iEventStruct);
00196
00197             // DEBUG
00198             std::ostream oEventStr;
00199             oEventStr << "Popped event: '"
00200                 << iEventStruct.describe() << "'.";
00201             STDAIR_LOG_DEBUG (oEventStr.str());
00202
00203             //
00204             return lProgressStatusSet;
00205         } catch (EventQueueException& lEventQueueException) {
00206             // DEBUG
00207             std::ostream oErrorMessage;
00208             oErrorMessage << "The event queue is empty: no event can be popped out.";
00209
00210             std::cerr << oErrorMessage.str() << std::endl;
00211             STDAIR_LOG_DEBUG(oErrorMessage.str());
00212         }
00213
00214         //
00215         return stdair::ProgressStatusSet(stdair::EventType::BKG_REQ);
00216     }
00217
00218     //
00219     // //////////////////////////////////////
00220     void EventQueueManager::run (EventQueue& ioEventQueue,
00221             stdair::EventStruct& iEventStruct) {
00222
00223         // Default event type
00224         stdair::EventType::EN_EventType lEventType = stdair::EventType::BKG_REQ;
00225
00226         // While no break point has been encountered, keep on extracting events
00227         while (ioEventQueue.isQueueDone() == false
00228             && lEventType != stdair::EventType::BRK_PT) {
00229             ioEventQueue.popEvent (iEventStruct);
00230             lEventType = iEventStruct.getEventType();
00231         }
00232     }
00233
00234     //
00235     // //////////////////////////////////////
00236     bool EventQueueManager::select (EventQueue& ioEventQueue,
00237             stdair::EventStruct& iEventStruct,
00238             const stdair::DateTime_T& iDateTime) {
00239
00240         // Search if an event has the given key
00241         const bool hasResearchBeenSuccessful =
00242             ioEventQueue.hasEventDateTime (iDateTime);
00243
00244         // If no event has the given key, return
00245         if (hasResearchBeenSuccessful == false) {
00246             return hasResearchBeenSuccessful;
00247         }
00248         assert (hasResearchBeenSuccessful == true);
00249
00250         // Default date time
00251         stdair::DateTime_T lDateTime = stdair::DEFAULT_EVENT_OLDEST_DATETIME;
00252
00253         // While the event with the given key has not been retrieved, keep on
00254         // extracting events
00255         while (ioEventQueue.isQueueDone() == false
00256             && lDateTime != iDateTime) {
00257             ioEventQueue.popEvent (iEventStruct);
00258             lDateTime = iEventStruct.getEventTime ();
00259         }
00260
00261         assert (lDateTime == iDateTime);
00262         return hasResearchBeenSuccessful;
00263     }

```

```

00274
00275 }
00276
00277 // //////////////////////////////////////
00278 void EventQueueManager::
00279 updateStatus (EventQueue& ioEventQueue,
00280              const stdair::EventType::EN_EventType& iEventType,
00281              const stdair::Count_T& iEventCount) {
00282
00286     ioEventQueue.updateStatus (iEventType, iEventCount);
00287 }
00288
00289 // //////////////////////////////////////
00290 void EventQueueManager::
00291 addStatus (EventQueue& ioEventQueue,
00292           const stdair::EventType::EN_EventType& iEventType,
00293           const stdair::Count_T& iEventCount) {
00294
00299     ioEventQueue.addStatus (iEventType, iEventCount);
00300 }
00301
00302 // //////////////////////////////////////
00303 bool EventQueueManager::
00304 isQueueDone (const EventQueue& iEventQueue) {
00305
00309     const bool isQueueDone = iEventQueue.isQueueDone();
00310
00311     //
00312     return isQueueDone;
00313 }
00314
00315 // //////////////////////////////////////
00316 const stdair::Count_T& EventQueueManager::
00317 getQueueSize (const EventQueue& iEventQueue) {
00318
00322     const stdair::Count_T& lQueueSize = iEventQueue.getQueueSize();
00323
00324     //
00325     return lQueueSize;
00326 }
00327
00328 // //////////////////////////////////////
00329 const stdair::Count_T& EventQueueManager::
00330 getExpectedTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue)
00331 {
00335     const stdair::Count_T& lExpectedTotalNumberOfEvents =
00336         ioEventQueue.getExpectedTotalNbOfEvents ();
00337
00338     //
00339     return lExpectedTotalNumberOfEvents;
00340 }
00341
00342 // //////////////////////////////////////
00343 const stdair::Count_T& EventQueueManager::
00344 getExpectedTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue,
00345                                              const
00346 stdair::EventType::EN_EventType& iEventType) {
00350     const stdair::Count_T& lExpectedTotalNumberOfEvents =
00351         ioEventQueue.getExpectedTotalNbOfEvents (iEventType);
00352
00353     //
00354     return lExpectedTotalNumberOfEvents;
00355 }
00356
00357 // //////////////////////////////////////
00358 const stdair::Count_T& EventQueueManager::
00359 getActualTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue) {
00360
00364     const stdair::Count_T& lActualTotalNumberOfEvents =
00365         ioEventQueue.getActualTotalNbOfEvents ();
00366
00367     //
00368     return lActualTotalNumberOfEvents;
00369 }
00370
00371 // //////////////////////////////////////
00372 const stdair::Count_T& EventQueueManager::
00373 getActualTotalNumberOfEventsToBeGenerated (const EventQueue& ioEventQueue,
00374                                           const
00375 stdair::EventType::EN_EventType& iEventType) {
00376
00380     const stdair::Count_T& lActualTotalNumberOfEvents =
00381         ioEventQueue.getActualTotalNbOfEvents (iEventType);
00382

```

```

00383     //
00384     return lActualTotalNumberOfEvents;
00385 }
00386 }
00387
00388 const stdair::ProgressStatus& EventQueueManager::
00389 getStatus (const EventQueue& iEventQueue,
00390           const stdair::EventType::EN_EventType& iEventType) {
00391
00392     return iEventQueue.getStatus(iEventType);
00393 }
00394
00395 const stdair::ProgressStatus& EventQueueManager::
00396 getStatus (const EventQueue& iEventQueue) {
00397
00398     return iEventQueue.getStatus();
00399 }
00400
00401 const stdair::ProgressStatus& EventQueueManager::
00402 getStatus (const EventQueue& iEventQueue) {
00403
00404     return iEventQueue.getStatus();
00405 }
00406
00407 }
00408
00409 }
00410
00411 }

```

23.41 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](#)

23.42 EventQueueManager.hpp

```

00001 #ifndef __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP
00002 #define __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010 #include <stdair/stdair_service_types.hpp>
00011 // SEVMgr
00012 #include <sevmgr/SEVMGR_Types.hpp>
00013
00014 // Forward declarations
00015 namespace stdair {
00016     struct ProgressStatusSet;
00017 }
00018
00019 namespace SEVMGR {
00020
00021     // Forward declarations
00022     class EventQueue;
00023
00024     class EventQueueManager : public stdair::CmdAbstract {
00025     friend class SEVMGR_Service;
00026
00027     private:
00028         // Business methodes
00029
00030     }
00031 }

```

```

00035     static void buildSampleQueue (stdair::STDAIR_ServicePtr_T,
00036                                   EventQueue&);
00037
00041     static stdair::BookingRequestStruct buildSampleBookingRequest (
00042         stdair::STDAIR_ServicePtr_T,
00043         const bool
00044         isForCRS = false);
00045
00047     static void reset (EventQueue&);
00048
00052     static void addEvent (EventQueue&, stdair::EventStruct&);
00053
00057     static const std::string describeKey (const EventQueue&);
00058
00062     static const std::string list (const EventQueue&);
00063
00067     static const std::string list (const EventQueue&,
00068                                   const stdair::EventType::EN_EventType&);
00069
00073     static stdair::ProgressStatusSet popEvent (EventQueue&,
00074                                                stdair::EventStruct&);
00075
00079     static void run (EventQueue&, stdair::EventStruct&);
00080
00097     static bool select (EventQueue&, stdair::EventStruct&, const
00098         stdair::DateTime_T&);
00099
00102     static void updateStatus (EventQueue&,
00103                              const stdair::EventType::EN_EventType&,
00104                              const stdair::Count_T&);
00109     static void addStatus (EventQueue&,
00110                           const stdair::EventType::EN_EventType&,
00111                           const stdair::Count_T&);
00112
00117     static bool hasProgressStatus (const EventQueue&,
00118                                   const stdair::EventType::EN_EventType&);
00119
00123     static bool isQueueDone (const EventQueue&);
00124
00125
00129     static const stdair::Count_T& getQueueSize(const EventQueue&);
00130
00134     static const stdair::Count_T&
00135     getExpectedTotalNumberOfEventsToBeGenerated(const EventQueue&);
00136
00140     static const stdair::Count_T&
00141     getExpectedTotalNumberOfEventsToBeGenerated(const EventQueue&,
00142                                                  const
00143     stdair::EventType::EN_EventType&);
00147
00148     static const stdair::Count_T&
00149     getActualTotalNumberOfEventsToBeGenerated(const EventQueue&);
00153
00154     static const stdair::Count_T&
00155     getActualTotalNumberOfEventsToBeGenerated(const EventQueue&,
00156                                                const
00157     stdair::EventType::EN_EventType&);
00160
00161     static const stdair::ProgressStatus& getStatus (const EventQueue&
00162     ,
00163     const
00164     stdair::EventType::EN_EventType&);
00167
00168     static const stdair::ProgressStatus& getStatus (const EventQueue&
00169     );
00170
00171 }
00172 #endif // __SEVMGR_CMD_EVENTQUEUEMANAGER_HPP

```

23.43 sevmgr/config/sevmgr-paths.hpp File Reference

Macros

- #define `PACKAGE` "sevmgr"
- #define `PACKAGE_NAME` "SEVMGR"
- #define `PACKAGE_VERSION` "1.00.0"
- #define `PREFIXDIR` "/usr"
- #define `EXEC_PREFIX` "/usr"

- `#define BINDIR "/usr/bin"`
- `#define LIBDIR "/usr/lib64"`
- `#define LIBEXECDIR "/usr/libexec"`
- `#define SBINDIR "/usr/sbin"`
- `#define SYSCONFDIR "/usr/etc"`
- `#define INCLUDEDIR "/usr/include"`
- `#define DATAROOTDIR "/usr/share"`
- `#define DATADIR "/usr/share"`
- `#define DOCDIR "/usr/share/doc/sevmgr-1.00.0"`
- `#define MANDIR "/usr/share/man"`
- `#define INFODIR "/usr/share/info"`
- `#define HTMLDIR "/usr/share/doc/sevmgr-1.00.0/html"`
- `#define PDFDIR "/usr/share/doc/sevmgr-1.00.0/html"`
- `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

23.43.1 Macro Definition Documentation

23.43.1.1 `#define PACKAGE "sevmgr"`

Definition at line 4 of file [sevmgr-paths.hpp](#).

23.43.1.2 `#define PACKAGE_NAME "SEVMGR"`

Definition at line 5 of file [sevmgr-paths.hpp](#).

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

23.43.1.3 `#define PACKAGE_VERSION "1.00.0"`

Definition at line 6 of file [sevmgr-paths.hpp](#).

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

23.43.1.4 `#define PREFIXDIR "/usr"`

Definition at line 7 of file [sevmgr-paths.hpp](#).

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

23.43.1.5 `#define EXEC_PREFIX "/usr"`

Definition at line 8 of file [sevmgr-paths.hpp](#).

23.43.1.6 `#define BINDIR "/usr/bin"`

Definition at line 9 of file [sevmgr-paths.hpp](#).

23.43.1.7 `#define LIBDIR "/usr/lib64"`

Definition at line 10 of file [sevmgr-paths.hpp](#).

23.43.1.8 `#define LIBEXECDIR "/usr/libexec"`

Definition at line 11 of file [sevmgr-paths.hpp](#).

23.43.1.9 `#define SBINDIR "/usr/sbin"`

Definition at line 12 of file [sevmgr-paths.hpp](#).

23.43.1.10 `#define SYSCONFDIR "/usr/etc"`

Definition at line 13 of file [sevmgr-paths.hpp](#).

23.43.1.11 `#define INCLUDEDIR "/usr/include"`

Definition at line 14 of file [sevmgr-paths.hpp](#).

23.43.1.12 `#define DATAROOTDIR "/usr/share"`

Definition at line 15 of file [sevmgr-paths.hpp](#).

23.43.1.13 `#define DATADIR "/usr/share"`

Definition at line 16 of file [sevmgr-paths.hpp](#).

23.43.1.14 `#define DOCDIR "/usr/share/doc/sevmgr-1.00.0"`

Definition at line 17 of file [sevmgr-paths.hpp](#).

23.43.1.15 `#define MANDIR "/usr/share/man"`

Definition at line 18 of file [sevmgr-paths.hpp](#).

23.43.1.16 `#define INFODIR "/usr/share/info"`

Definition at line 19 of file [sevmgr-paths.hpp](#).

23.43.1.17 `#define HTMLDIR "/usr/share/doc/sevmgr-1.00.0/html"`

Definition at line 20 of file [sevmgr-paths.hpp](#).

23.43.1.18 `#define PDFDIR "/usr/share/doc/sevmgr-1.00.0/html"`

Definition at line 21 of file [sevmgr-paths.hpp](#).

23.43.1.19 `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

Definition at line 22 of file [sevmgr-paths.hpp](#).

23.44 sevmgr-paths.hpp

```
00001 #ifndef __SEVMGR_PATHS_HPP__
00002 #define __SEVMGR_PATHS_HPP__
00003
00004 #define PACKAGE "sevmgr"
00005 #define PACKAGE_NAME "SEVMGR"
00006 #define PACKAGE_VERSION "1.00.0"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib64"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/sevmgr-1.00.0"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/sevmgr-1.00.0/html"
00021 #define PDFDIR "/usr/share/doc/sevmgr-1.00.0/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __SEVMGR_PATHS_HPP__
```

23.45 sevmgr/config/sevmgr-paths.hpp.in File Reference

Macros

- `#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@"`

23.45.1 Macro Definition Documentation

23.45.1.1 `#define __SEVMGR_PATHS_HPP__`

Definition at line 2 of file [sevmgr-paths.hpp.in](#).

23.45.1.2 `#define PACKAGE "@PACKAGE@"`

Definition at line 4 of file [sevmgr-paths.hpp.in](#).

23.45.1.3 `#define PACKAGE_NAME "@PACKAGE_NAME@"`

Definition at line 5 of file [sevmgr-paths.hpp.in](#).

23.45.1.4 `#define PACKAGE_VERSION "@PACKAGE_VERSION@"`

Definition at line 6 of file [sevmgr-paths.hpp.in](#).

23.45.1.5 `#define PREFIXDIR "@prefix@"`

Definition at line 7 of file [sevmgr-paths.hpp.in](#).

23.45.1.6 `#define EXEC_PREFIX "@exec_prefix@"`

Definition at line 8 of file [sevmgr-paths.hpp.in](#).

23.45.1.7 `#define BINDIR "@bindir@"`

Definition at line 9 of file [sevmgr-paths.hpp.in](#).

23.45.1.8 `#define LIBDIR "@libdir@"`

Definition at line 10 of file [sevmgr-paths.hpp.in](#).

23.45.1.9 `#define LIBEXECDIR "@libexecdir@"`

Definition at line 11 of file [sevmgr-paths.hpp.in](#).

23.45.1.10 `#define SBINDIR "@sbindir@"`

Definition at line 12 of file [sevmgr-paths.hpp.in](#).

23.45.1.11 `#define SYSCONFDIR "@sysconfdir@"`

Definition at line 13 of file [sevmgr-paths.hpp.in](#).

23.45.1.12 `#define INCLUDEDIR "@includedir@"`

Definition at line 14 of file [sevmgr-paths.hpp.in](#).

23.45.1.13 `#define DATAROOTDIR "@datarootdir@"`

Definition at line 15 of file [sevmgr-paths.hpp.in](#).

23.45.1.14 `#define DATADIR "@datadir@"`

Definition at line 16 of file [sevmgr-paths.hpp.in](#).

23.45.1.15 `#define DOCDIR "@docdir@"`

Definition at line 17 of file [sevmgr-paths.hpp.in](#).

23.45.1.16 `#define MANDIR "@mandir@"`

Definition at line 18 of file [sevmgr-paths.hpp.in](#).

23.45.1.17 `#define INFODIR "@infodir@"`

Definition at line 19 of file [sevmgr-paths.hpp.in](#).

23.45.1.18 `#define HTMLDIR "@htmldir@"`

Definition at line 20 of file [sevmgr-paths.hpp.in](#).

23.45.1.19 `#define PDFDIR "@pdfdir@"`

Definition at line 21 of file [sevmgr-paths.hpp.in](#).

23.45.1.20 `#define STDAIR_SAMPLE_DIR "@sampledir@"`

Definition at line 22 of file [sevmgr-paths.hpp.in](#).

23.46 sevmgr-paths.hpp.in

```
00001 #ifndef __SEVMGR_PATHS_HPP__
00002 #define __SEVMGR_PATHS_HPP__
00003
00004 #define PACKAGE "@PACKAGE@"
00005 #define PACKAGE_NAME "@PACKAGE_NAME@"
00006 #define PACKAGE_VERSION "@PACKAGE_VERSION@"
00007 #define PREFIXDIR "@prefix@"
00008 #define EXEC_PREFIX "@exec_prefix@"
00009 #define BINDIR "@bindir@"
00010 #define LIBDIR "@libdir@"
00011 #define LIBEXECDIR "@libexecdir@"
00012 #define SBINDIR "@sbindir@"
00013 #define SYSCONFDIR "@sysconfdir@"
00014 #define INCLUDEDIR "@includedir@"
00015 #define DATAROOTDIR "@datarootdir@"
00016 #define DATADIR "@datadir@"
```

```

00017 #define DOCDIR "@docdir@"
00018 #define MANDIR "@mandir@"
00019 #define INFODIR "@infodir@"
00020 #define HTMLDIR "@htmldir@"
00021 #define PDFDIR "@pdfdir@"
00022 #define STDAIR_SAMPLE_DIR "@sampledir@"
00023
00024 #endif // __SEVMGR_PATHS_HPP__

```

23.47 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](#)

23.48 FacSEVMGRServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // Sevmgr
00009 #include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
00010 #include <sevmgr/service/SEVMGR_ServiceContext.hpp>
00011
00012 namespace SEVMGR {
00013
00014     FacSEVMGRServiceContext* FacSEVMGRServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacSEVMGRServiceContext::~FacSEVMGRServiceContext
00018     () {
00019         _instance = NULL;
00020     }
00021
00022     // //////////////////////////////////////
00023     FacSEVMGRServiceContext&
00024     FacSEVMGRServiceContext::instance () {
00025         if (_instance == NULL) {
00026             _instance = new FacSEVMGRServiceContext ();
00027             assert (_instance != NULL);
00028             stdair::FacSupervisor::instance().
00029             registerServiceFactory (_instance);
00030             return *_instance;
00031         }
00032
00033         // //////////////////////////////////////
00034         SEVMGR_ServiceContext& FacSEVMGRServiceContext::create
00035         () {
00036             SEVMGR_ServiceContext* aServiceContext_ptr = NULL;
00037             aServiceContext_ptr = new SEVMGR_ServiceContext ();
00038             assert (aServiceContext_ptr != NULL);
00039
00040             // The new object is added to the Bom pool
00041             _pool.push_back (aServiceContext_ptr);
00042             return *aServiceContext_ptr;
00043         }
00044     }
00045
00046 }

```

23.49 sevmgr/factory/FacSEVMGRServiceContext.hpp File Reference

```
#include <stdair/service/FacServiceAbstract.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class [SEVMGR::FacSEVMGRServiceContext](#)

Namespaces

- namespace [SEVMGR](#)

23.50 FacSEVMGRServiceContext.hpp

```
00001 #ifndef __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP
00002 #define __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009 // Sevmgr
00010 #include <sevmgr/SEVMGR_Types.hpp>
00011
00012 namespace SEVMGR {
00013
00015     class SEVMGR_ServiceContext;
00016
00018     class FacSEVMGRServiceContext : public
stdair::FacServiceAbstract {
00019     public:
00020
00024         static FacSEVMGRServiceContext& instance ();
00025
00030         ~FacSEVMGRServiceContext ();
00031
00035         SEVMGR_ServiceContext& create ();
00036
00037     protected:
00038         FacSEVMGRServiceContext () {}
00043
00044     private:
00046         static FacSEVMGRServiceContext* _instance;
00047     };
00048
00049 }
00050 #endif // __SEVMGR_FAC_FACSEVMGRSERVICECONTEXT_HPP
```

23.51 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)

23.51.1 Function Documentation

23.51.1.1 BOOST_PYTHON_MODULE (libpysevmgr)

Definition at line 152 of file [pysevmgr.cpp](#).

References [SEVMGR::PYEventQueueManager::init\(\)](#), and [SEVMGR::PYEventQueueManager::sevmgr\(\)](#).

23.52 pysevmgr.cpp

```
00001 // STL
00002 #include <cassert>
00003 #include <stdexcept>
00004 #include <fstream>
00005 #include <sstream>
00006 #include <string>
00007 #include <list>
00008 #include <vector>
00009 // Boost String
00010 #include <boost/python.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/stdair_exceptions.hpp>
00014 #include <stdair/basic/BasFileMgr.hpp>
00015 #include <stdair/basic/BasLogParams.hpp>
00016 #include <stdair/basic/BasDBParams.hpp>
00017 // SEvMgr
00018 #include <sevmgr/SEVMGR_Service.hpp>
00019
00020 namespace SEVMGR {
00021
00022     struct PYEventQueueManager {
00023     public:
```

```

00025     std::string sevmgr() {
00026         std::ostringstream oStream;
00027
00028         // Sanity check
00029         if (_logOutputStream == NULL) {
00030             oStream << "The log filepath is not valid." << std::endl;
00031             return oStream.str();
00032         }
00033         assert (_logOutputStream != NULL);
00034
00035         try {
00036
00037             // DEBUG
00038             *_logOutputStream << "Default service" << std::endl;
00039
00040             if (_sevmgrService == NULL) {
00041                 oStream << "The Sevmgr service has not been initialised, "
00042                     << "i.e., the init() method has not been called "
00043                     << "correctly on the PYEventQueueManager object. Please "
00044                     << "check that all the parameters are not empty and "
00045                     << "point to actual files.";
00046                 *_logOutputStream << oStream.str();
00047                 return oStream.str();
00048             }
00049             assert (_sevmgrService != NULL);
00050
00051             // Do the sevmgr
00052             _sevmgrService->buildSampleBom();
00053
00054             // DEBUG
00055             *_logOutputStream << "Default service returned" << std::endl;
00056
00057             // DEBUG
00058             *_logOutputStream << "Sevmgr output: " << oStream.str() << std::endl;
00059
00060         } catch (const stdair::RootException& eSevmgrError) {
00061             *_logOutputStream << "Sevmgr error: " << eSevmgrError.what()
00062                 << std::endl;
00063
00064         } catch (const std::exception& eStdError) {
00065             *_logOutputStream << "Error: " << eStdError.what() << std::endl;
00066
00067         } catch (...) {
00068             *_logOutputStream << "Unknown error" << std::endl;
00069         }
00070
00071         return oStream.str();
00072     }
00073
00074     public:
00076     PYEventQueueManager() : _sevmgrService (NULL),
00077         _logOutputStream (NULL) {
00078     }
00080     PYEventQueueManager (const PYEventQueueManager
00081 & iPYEventQueueManager)
00082     : _sevmgrService (iPYEventQueueManager._sevmgrService),
00083       _logOutputStream (iPYEventQueueManager._logOutputStream) {
00084     }
00086     ~PYEventQueueManager() {
00087         _sevmgrService = NULL;
00088         _logOutputStream = NULL;
00089     }
00090
00092     bool init (const std::string& iLogFilePath,
00093               const std::string& iDBUser, const std::string& iDBPasswd,
00094               const std::string& iDBHost, const std::string& iDBPort,
00095               const std::string& iDBDBName) {
00096         bool isEverythingOK = true;
00097
00098         try {
00099
00100             // Check that the file path given as input corresponds to an actual
00101             file
00102             const bool isWriteable = (iLogFilePath.empty() == false);
00103             // stdair::BasFileMgr::isWriteable (iLogFilePath);
00104             if (isWriteable == false) {
00105                 isEverythingOK = false;
00106                 return isEverythingOK;
00107             }
00108
00109             // Set the log parameters
00110             _logOutputStream = new std::ofstream;
00111             assert (_logOutputStream != NULL);
00112
00113             // Open and clean the log outputfile

```

```

00113     _logOutputStream->open (iLogFilepath.c_str());
00114     _logOutputStream->clear();
00115
00116     // DEBUG
00117     *_logOutputStream << "Python wrapper initialisation" << std::endl;
00118     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00119                                           *_logOutputStream);
00120
00121     // Initialise the context
00122     stdair::BasDBParams lDBParams (iDBUser, iDBPasswd, iDBHost, iDBPort,
00123                                   iDBDBName);
00124     _sevmgrService = new SEVMGR_Service (lLogParams,
00125                                         lDBParams);
00126
00127     // DEBUG
00128     *_logOutputStream << "Python wrapper initialised" << std::endl;
00129
00130     } catch (const stdair::RootException& eSevmgrError) {
00131         *_logOutputStream << "Sevmgr error: " << eSevmgrError.what()
00132         << std::endl;
00133     } catch (const std::exception& eStdError) {
00134         *_logOutputStream << "Error: " << eStdError.what() << std::endl;
00135     } catch (...) {
00136         *_logOutputStream << "Unknown error" << std::endl;
00137     }
00138
00139     return isEverythingOK;
00140 }
00141
00142 private:
00143     SEVMGR_Service* _sevmgrService;
00144     std::ofstream* _logOutputStream;
00145 };
00146
00147 }
00148
00149 }
00150
00151 // //////////////////////////////////////
00152 BOOST_PYTHON_MODULE(libpysevmgr) {
00153     boost::python::class_<SEVMGR::PYEventQueueManager> ("PYEventQueueManager")
00154         .def ("sevmgr", &SEVMGR::PYEventQueueManager::sevmgr
00155         )
00156         .def ("init", &SEVMGR::PYEventQueueManager::init
00157         );
00158 }

```

23.53 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](#)

23.54 SEVMGR_Service.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/basic/BasChronometer.hpp>
00011 #include <stdair/basic/BasConst_General.hpp>
00012 #include <stdair/basic/JSonCommand.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/BomDisplay.hpp>
00015 #include <stdair/bom/EventStruct.hpp>
00016 #include <stdair/bom/BookingRequestStruct.hpp>
00017 #include <stdair/bom/BomJSONImport.hpp>
00018 #include <stdair/service/Logger.hpp>
00019 #include <stdair/STDAIR_Service.hpp>
00020 // Sevmgr
00021 #include <sevmgr/basic/BasConst_SEVMGR_Service.hpp>
00022 >
00023 #include <sevmgr/factory/FacSEVMGRServiceContext.hpp>
00024 >
00025 #include <sevmgr/command/EventQueueManager.hpp>
00026 >
00027 #include <sevmgr/service/SEVMGR_ServiceContext.hpp>
00028 >
00029 #include <sevmgr/SEVMGR_Service.hpp>
00030 #include <sevmgr/bom/EventQueue.hpp>
00031 #include <sevmgr/bom/BomJSONExport.hpp>
00032 namespace SEVMGR {
00033 // //////////////////////////////////////
00034 SEVMGR_Service::SEVMGR_Service() : _sevmgrServiceContext (NULL) {
00035     assert (false);
00036 }
00037 // //////////////////////////////////////
00038 SEVMGR_Service::SEVMGR_Service (const SEVMGR_Service& iService)
00039 : _sevmgrServiceContext (NULL) {
00040     assert (false);
00041 }
00042 // //////////////////////////////////////
00043 SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams& iLogParams,
00044                                 const stdair::BasDBParams& iDBParams)
00045 : _sevmgrServiceContext (NULL) {
00046     // Initialise the STDAIR service handler
00047     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00048         initStdAirService (iLogParams, iDBParams);
00049     // Initialise the service context
00050     initServiceContext();
00051     // Add the StdAir service context to the SEvMgr service context
00052     // \note SEvMgr owns the STDAIR service resources here.
00053     const bool ownStdairService = true;
00054     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00055     // Initialise the (remaining of the) context
00056     initSevmgrService();
00057 }
00058 // //////////////////////////////////////
00059 SEVMGR_Service::SEVMGR_Service (const stdair::BasLogParams& iLogParams)
00060 : _sevmgrServiceContext (NULL) {
00061     // Initialise the STDAIR service handler
00062     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00063         initStdAirService (iLogParams);
00064     // Initialise the service context
00065     initServiceContext();
00066 }

```

```

00074 // Add the StdAir service context to the SEvMgr service context
00075 // \note SEvMgr owns the STDAIR service resources here.
00076 const bool ownStdairService = true;
00077 addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00078
00079 // Initialise the (remaining of the) context
00080 initSevmgrService();
00081 }
00082
00083 // //////////////////////////////////////
00084 SEVMGR_Service::
00085 SEVMGR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00086 : _sevmgrServiceContext (NULL) {
00087
00088 // Initialise the service context
00089 initServiceContext();
00090
00091 // Add the StdAir service context to the SEvMgr service context
00092 // \note SEvMgr does not own the STDAIR service resources here.
00093 const bool doesNotOwnStdairService = false;
00094 addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00095
00096 // Initialise the context
00097 initSevmgrService();
00098 }
00099
00100 // //////////////////////////////////////
00101 SEVMGR_Service::~SEVMGR_Service() {
00102 // Delete/Clean all the objects from memory
00103 finalise();
00104 }
00105
00106 // //////////////////////////////////////
00107 void SEVMGR_Service::finalise() {
00108 assert (_sevmgrServiceContext != NULL);
00109 // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00110 _sevmgrServiceContext->reset();
00111 }
00112
00113 // //////////////////////////////////////
00114 void SEVMGR_Service::initServiceContext() {
00115 // Initialise the service context
00116 SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00117 FacSEVMGRServiceContext::instance().
00118 create();
00119 _sevmgrServiceContext = &lSEVMGR_ServiceContext;
00120 }
00121
00122 // //////////////////////////////////////
00123 void SEVMGR_Service::
00124 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00125 const bool iOwnStdairService) {
00126 // Retrieve the SEvMgr service context
00127 assert (_sevmgrServiceContext != NULL);
00128 SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00129 *_sevmgrServiceContext;
00130
00131 // Store the STDAIR service object within the (SEvMgr) service context
00132 lSEVMGR_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00133 iOwnStdairService);
00134 }
00135
00136 // //////////////////////////////////////
00137 stdair::STDAIR_ServicePtr_T SEVMGR_Service::
00138 initStdAirService (const stdair::BasLogParams& iLogParams,
00139 const stdair::BasDBParams& iDBParams) {
00140
00141 stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00142 boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00143 assert (lSTDAIR_Service_ptr != NULL);
00144
00145 return lSTDAIR_Service_ptr;
00146 }
00147
00148 // //////////////////////////////////////
00149 stdair::STDAIR_ServicePtr_T SEVMGR_Service::
00150 initStdAirService (const stdair::BasLogParams& iLogParams) {
00151
00152 stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00153 boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00154 assert (lSTDAIR_Service_ptr != NULL);
00155
00156 return lSTDAIR_Service_ptr;
00157 }
00158
00159 // //////////////////////////////////////
00160 void SEVMGR_Service::initSevmgrService() {

```



```

00170     // Do nothing at this stage. A sample BOM tree may be built by
00171     // calling the buildSampleBom() method
00172 }
00173
00174 // //////////////////////////////////////
00175 void SEVMGR_Service::buildSampleQueue() {
00176     // Retrieve the SEvMgr service context
00177     if (_sevmgrServiceContext == NULL) {
00178         throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
00179             "not been initialised");
00180     }
00181     assert (_sevmgrServiceContext != NULL);
00182     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
00183     _sevmgrServiceContext;
00184
00185     // Retrieve the StdAir service context
00186     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00187     lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();
00188
00189     // Retrieve the EventQueue
00190     EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue();
00191
00192     // Delegate the building process to the dedicated command
00193     EventQueueManager::buildSampleQueue (
00194     lSTDAIR_Service_ptr, lEventQueue);
00195 }
00196
00197 // //////////////////////////////////////
00198 stdair::BookingRequestStruct SEVMGR_Service::
00199 buildSampleBookingRequest(const bool isForCRS) {
00200     // Retrieve the SEvMgr service context
00201     if (_sevmgrServiceContext == NULL) {
00202         throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
00203             "not been initialised");
00204     }
00205     assert (_sevmgrServiceContext != NULL);
00206     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
00207     _sevmgrServiceContext;
00208
00209     // Retrieve the StdAir service context
00210     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00211     lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();
00212
00213     // Delegate the booking request building to the dedicated service
00214     stdair::BookingRequestStruct oBookingRequest =
00215     EventQueueManager::buildSampleBookingRequest
00216     (lSTDAIR_Service_ptr,
00217         isForCRS);
00218     return oBookingRequest;
00219 }
00220
00221 // //////////////////////////////////////
00222 std::string SEVMGR_Service::describeKey() const {
00223     // Retrieve the SEvMgr service context
00224     if (_sevmgrServiceContext == NULL) {
00225         throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
00226             "not been initialised");
00227     }
00228     assert (_sevmgrServiceContext != NULL);
00229     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
00230     _sevmgrServiceContext;
00231
00232     // Retrieve the event queue
00233     EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue();
00234
00235     // Delegate the key display to the dedicated command
00236     return EventQueueManager::describeKey(
00237     lEventQueue);
00238 }
00239
00240 // //////////////////////////////////////
00241 std::string SEVMGR_Service::list() const {
00242     // Retrieve the SEvMgr service context
00243     if (_sevmgrServiceContext == NULL) {
00244         throw stdair::NonInitialisedServiceException ("The SEvMgr service has "
00245             "not been initialised");
00246     }
00247     assert (_sevmgrServiceContext != NULL);

```

```

00251
00252     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
00253     _sevmgrServiceContext;
00254
00255     // Retrieve the event queue
00256     EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue ()
00257 ;
00258
00259     // Delegate the key display to the dedicated command
00260     return EventQueueManager::list (lEventQueue);
00261 }
00262
00263 // //////////////////////////////////////
00264 std::string SEVMGR_Service::
00265 list (const stdair::EventType& iEventType) const {
00266
00267     // Retrieve the SEVMgr service context
00268     if (_sevmgrServiceContext == NULL) {
00269         throw stdair::NonInitialisedServiceException ("The SEVMgr service has "
00270             "not been initialised");
00271     }
00272     assert (_sevmgrServiceContext != NULL);
00273
00274     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
00275     _sevmgrServiceContext;
00276
00277     // Retrieve the event queue
00278     EventQueue& lEventQueue = lSEVMGR_ServiceContext.getEventQueue ()
00279 ;
00280
00281     // Delegate the key display to the dedicated command
00282     return EventQueueManager::list (lEventQueue,
00283         iEventType);
00284 }
00285
00286 // //////////////////////////////////////
00287 std::string SEVMGR_Service::
00288 jsonHandler (const stdair::JSONString& iJSONString) const {
00289
00290     //
00291     // Extract from the JSON-ified string the command
00292     //
00293     stdair::JJsonCommand::EN_JJsonCommand lEN_JJsonCommand;
00294     const bool hasCommandBeenRetrieved =
00295         stdair::BomJSONImport::jsonImportCommand (iJSONString,
00296             lEN_JJsonCommand);
00297
00298     if (hasCommandBeenRetrieved == false) {
00299         // Return an error JSON-ified string
00300         std::ostringstream oErrorMessage;
00301         oErrorMessage << "{\"error\": \"Wrong JSON-ified string: \"
00302             << "the command is not understood.\"}";
00303         return oErrorMessage.str();
00304     }
00305     assert (hasCommandBeenRetrieved == true);
00306
00307     //
00308     // Dispatch the command to the right JJson service handler
00309     //
00310     switch (lEN_JJsonCommand) {
00311     case stdair::JJsonCommand::EVENT_LIST:{
00312
00313         //
00314         // Try to extract the event type from the JSON-ified string
00315         //
00316         stdair::EventType::EN_EventType lEN_EventType;
00317         const bool hasEventTypeBeenRetrieved =
00318             stdair::BomJSONImport::jsonImportEventType (iJSONString,
00319                 lEN_EventType);
00320
00321         if (hasEventTypeBeenRetrieved == true) {
00322             return jsonExportEventQueue (lEN_EventType);
00323         }
00324         return jsonExportEventQueue ();
00325     }
00326     default: {
00327         // Return an Error string
00328         std::ostringstream lErrorCmdMessage;
00329         const std::string& lCommandStr =
00330             stdair::JJsonCommand::getLabel (lEN_JJsonCommand);
00331         lErrorCmdMessage << "{\"error\": \"The command '\" << lCommandStr
00332             << "' is not handled by the DSIm service.\"}";
00333         return lErrorCmdMessage.str();
00334         break;
00335     }
00336 }
00337 }
00338
00339 }
00340
00341 }
00342

```

```

00333     // Return an error JSON-ified string
00334     assert (false);
00335     std::string lJSONDump ("{\"error\": \"Wrong JSON-ified string\"}");
00336     return lJSONDump;
00337 }
00338 }
00339
00340 // //////////////////////////////////////
00341 std::string SEVMGR_Service::
00342 jsonExportEventQueue (const
stdair::EventType::EN_EventType& iEventType) const {
00343
00344     std::ostringstream oStr;
00345
00346     // Retrieve the SEvMgr service context
00347     if (_sevmgrServiceContext == NULL) {
00348         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00349             "has not been initialised")
;
00350     }
00351     assert (_sevmgrServiceContext != NULL);
00352
00353     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00354
00355     // Retrieve the StdAir service context
00356     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00357         lSEVMGR_ServiceContext.getSTDAIR_ServicePtr();
00358
00359     // Retrieve the event queue
00360     const EventQueue& lEventQueue =
00361         lSEVMGR_ServiceContext.getEventQueue();
00362
00363     // Delegate the JSON export to the dedicated command
00364     BomJSONExport::jsonExportEventQueue (
lSTDAIR_Service_ptr, oStr,
00365                                     lEventQueue, iEventType);
00366     return oStr.str();
00367 }
00368 }
00369
00370 // //////////////////////////////////////
00371 std::string SEVMGR_Service::
00372 jsonExportEvent (const stdair::EventStruct& iEvent) const {
00373
00374     std::ostringstream oStr;
00375
00376     // Retrieve the SEvMgr service context
00377     if (_sevmgrServiceContext == NULL) {
00378         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00379             "has not been initialised")
;
00380     }
00381     assert (_sevmgrServiceContext != NULL);
00382
00383     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00384
00385     // Retrieve the STDAIR service object from the (SEvMgr) service context
00386     stdair::STDAIR_Service& lSTDAIR_Service =
00387         lSEVMGR_ServiceContext.getSTDAIR_Service();
00388
00389     // Delegate the JSON export to the dedicated service
00390     oStr << lSTDAIR_Service.jsonExportEventObject (iEvent);
00391
00392     return oStr.str();
00393 }
00394 }
00395
00396 // //////////////////////////////////////
00397 stdair::ProgressStatusSet SEVMGR_Service::
00398 popEvent (stdair::EventStruct& iEventStruct) const {
00399
00400     // Retrieve the SEvMgr service context
00401     if (_sevmgrServiceContext == NULL) {
00402         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00403             "has not been initialised")
;
00404     }
00405     assert (_sevmgrServiceContext != NULL);
00406     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00407
00408     // Retrieve the event queue object instance
00409     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00410

```

```

00411     // Delegate the call to the dedicated command
00412     return EventQueueManager::popEvent(lQueue,
iEventStruct);
00413 }
00414
00415 // //////////////////////////////////////
00416 void SEVMGR_Service::
00417 run (stdair::EventStruct& iEventStruct) const {
00418
00419     // Retrieve the SEvMgr service context
00420     if (_sevmgrServiceContext == NULL) {
00421         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00422                                                         "has not been initialised")
;
00423     }
00424     assert (_sevmgrServiceContext != NULL);
00425     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00426
00427     // Retrieve the event queue object instance
00428     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00429
00430     // Delegate the call to the dedicated command
00431     EventQueueManager::run (lQueue, iEventStruct);
00432
00433 }
00434
00435 // //////////////////////////////////////
00436 bool SEVMGR_Service::
00437 select (stdair::EventStruct& iEventStruct,
00438         const stdair::DateTime_T& iEventDateTime) const {
00439
00440     // Retrieve the SEvMgr service context
00441     if (_sevmgrServiceContext == NULL) {
00442         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00443                                                         "has not been initialised")
;
00444     }
00445     assert (_sevmgrServiceContext != NULL);
00446     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00447
00448     // Retrieve the event queue object instance
00449     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00450
00451     // Delegate the call to the dedicated command
00452     return EventQueueManager::select (lQueue,
iEventStruct, iEventDateTime);
00453
00454 }
00455
00456 // //////////////////////////////////////
00457 void SEVMGR_Service::
00458 updateStatus (const stdair::EventType::EN_EventType& iEventType
,
00459              const stdair::Count_T& iEventCount) const {
00460
00461     // Retrieve the SEvMgr service context
00462     if (_sevmgrServiceContext == NULL) {
00463         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00464                                                         "has not been initialised")
;
00465     }
00466     assert (_sevmgrServiceContext != NULL);
00467     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
_sevmgrServiceContext;
00468
00469     // Retrieve the event queue object instance
00470     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00471
00472     // Delegate the call to the dedicated command
00473     EventQueueManager::updateStatus (lQueue,
iEventType, iEventCount);
00474
00475 }
00476
00477 // //////////////////////////////////////
00478 void SEVMGR_Service::
00479 addStatus (const stdair::EventType::EN_EventType& iEventType,
00480           const stdair::Count_T& iEventCount) const {
00481
00482     // Retrieve the SEvMgr service context
00483     if (_sevmgrServiceContext == NULL) {
00484         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00485                                                         "has not been initialised")
;
00486     }
00487     assert (_sevmgrServiceContext != NULL);

```

```

00487     SEVMGR_ServiceContext& lSEVMGR_ServiceContext = *
    _sevmgrServiceContext;
00488
00489     // Retrieve the event queue object instance
00490     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00491
00492     // Delegate the call to the dedicated function
00493     EventQueueManager::addStatus (lQueue,
    iEventType, iEventCount);
00494 }
00495
00496 // //////////////////////////////////////
00497 bool SEVMGR_Service::isQueueDone() const {
00498
00499     // Retrieve the SEvMgr service context
00500     if (_sevmgrServiceContext == NULL) {
00501         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
    "has not been initialised")
00502     ;
00503     }
00504     assert (_sevmgrServiceContext != NULL);
00505     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;
00506
00507     // Retrieve the event queue object instance
00508     const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
    ;
00509
00510     // Calculates whether the event queue has been fully emptied
00511     const bool isQueueDone = EventQueueManager::isQueueDone
    (lQueue);
00512
00513     //
00514     return isQueueDone;
00515 }
00516
00517 // //////////////////////////////////////
00518 bool SEVMGR_Service::hasProgressStatus(const
    stdair::EventType::EN_EventType& iEventType) const {
00519
00520     // Retrieve the SEvMgr service context
00521     if (_sevmgrServiceContext == NULL) {
00522         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
    "has not been initialised")
00523     ;
00524     }
00525     assert (_sevmgrServiceContext != NULL);
00526     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;
00527
00528     // Retrieve the event queue object instance
00529     const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
    ;
00530
00531     // Calculates whether the event queue has been fully emptied
00532     const bool hasProgressStatus =
    EventQueueManager::hasProgressStatus
    (lQueue, iEventType);
00533
00534     //
00535     return hasProgressStatus;
00536 }
00537
00538 // //////////////////////////////////////
00539 const stdair::Count_T& SEVMGR_Service::getQueueSize
    () const {
00540
00541     // Retrieve the SEvMgr service context
00542     if (_sevmgrServiceContext == NULL) {
00543         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
    "has not been initialised")
00544     ;
00545     }
00546     assert (_sevmgrServiceContext != NULL);
00547     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
    *_sevmgrServiceContext;
00548
00549     // Retrieve the event queue object instance
00550     const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
    ;
00551
00552     // Delegate the call to the dedicated command
00553     return EventQueueManager::getQueueSize(
    lQueue);
00554 }
00555
00556 // //////////////////////////////////////
00557
00558
00559
00560

```

```

00561 void SEVMGR_Service::reset() const {
00562
00563     // Retrieve the SEvMgr service context
00564     if (_sevmgrServiceContext == NULL) {
00565         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00566                                                         "has not been initialised")
00567     };
00568     assert (_sevmgrServiceContext != NULL);
00569     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00570         *_sevmgrServiceContext;
00571
00572     // Retrieve the event queue object instance
00573     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00574
00575     // Delegate the call to the dedicated command
00576     EventQueueManager::reset (lQueue);
00577 }
00578
00579 // //////////////////////////////////////
00580 EventQueue& SEVMGR_Service::getEventQueue
00581 () const {
00582     // Retrieve the SEvMgr service context
00583     if (_sevmgrServiceContext == NULL) {
00584         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00585                                                         "has not been initialised")
00586     };
00587     assert (_sevmgrServiceContext != NULL);
00588
00589     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00590         *_sevmgrServiceContext;
00591
00592     return lSEVMGR_ServiceContext.getEventQueue();
00593 }
00594
00595 // //////////////////////////////////////
00596 void SEVMGR_Service::addEvent(stdair::EventStruct&
iEventStruct) const {
00597     // Retrieve the SEvMgr service context
00598     if (_sevmgrServiceContext == NULL) {
00599         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00600                                                         "has not been initialised")
00601     };
00602     assert (_sevmgrServiceContext != NULL);
00603     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00604         *_sevmgrServiceContext;
00605
00606     // Retrieve the event queue object instance
00607     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00608
00609     // Delegate the call to the dedicated command
00610     EventQueueManager::addEvent (lQueue,
iEventStruct);
00611 }
00612
00613 // //////////////////////////////////////
00614 const stdair::Count_T& SEVMGR_Service::
00615 getExpectedTotalNumberOfEventsToBeGenerated
00616 () const {
00617     // Retrieve the SEvMgr service context
00618     if (_sevmgrServiceContext == NULL) {
00619         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00620                                                         "has not been initialised")
00621     };
00622     assert (_sevmgrServiceContext != NULL);
00623     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00624         *_sevmgrServiceContext;
00625
00626     // Retrieve the event queue object instance
00627     const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
00628 ;
00629
00630     // Delegate the call to the dedicated function
00631     return EventQueueManager::getExpectedTotalNumberOfEventsToBeGenerated
(lQueue);
00632 }
00633
00634 // //////////////////////////////////////
00635 const stdair::Count_T& SEVMGR_Service::
00636 getExpectedTotalNumberOfEventsToBeGenerated
00637 (const stdair::EventType::EN_EventType& iEventType) const {

```

```

00637
00638 // Retrieve the SEvMgr service context
00639 if (_sevmgrServiceContext == NULL) {
00640     throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00641                                                    "has not been initialised")
00642 ;
00643 }
00644 assert (_sevmgrServiceContext != NULL);
00645 SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00646 *_sevmgrServiceContext;
00647 // Retrieve the event queue object instance
00648 const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
00649 ;
00650 // Delegate the call to the dedicated function
00651 return EventQueueManager::getExpectedTotalNumberOfEventsToBeGenerated
00652 (lQueue,
00653 iEventType);
00654 }
00655 // //////////////////////////////////////
00656 const stdair::Count_T& SEVMGR_Service::
00657 getActualTotalNumberOfEventsToBeGenerated
00658 () const {
00659     // Retrieve the SEvMgr service context
00660     if (_sevmgrServiceContext == NULL) {
00661         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00662                                                        "has not been initialised")
00663     ;
00664     }
00665     assert (_sevmgrServiceContext != NULL);
00666     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00667 *_sevmgrServiceContext;
00668 // Retrieve the event queue object instance
00669 const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
00670 ;
00671 // Delegate the call to the dedicated function
00672 return EventQueueManager::getActualTotalNumberOfEventsToBeGenerated
00673 (lQueue);
00674 }
00675 // //////////////////////////////////////
00676 const stdair::Count_T& SEVMGR_Service::
00677 getActualTotalNumberOfEventsToBeGenerated
00678 (const stdair::EventType::EN_EventType& iEventType) const {
00679     // Retrieve the SEvMgr service context
00680     if (_sevmgrServiceContext == NULL) {
00681         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00682                                                        "has not been initialised")
00683     ;
00684     }
00685     assert (_sevmgrServiceContext != NULL);
00686     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00687 *_sevmgrServiceContext;
00688 // Retrieve the event queue object instance
00689 const EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue()
00690 ;
00691 // Delegate the call to the dedicated function
00692 return EventQueueManager::getActualTotalNumberOfEventsToBeGenerated
00693 (lQueue,
00694 iEventType);
00695 }
00696 }
00697
00698 const stdair::STDAIR_Service& SEVMGR_Service::getSTDAIR_Service() const {
00699
00700     // Retrieve the StdAir service context
00701     if (_sevmgrServiceContext == NULL) {
00702         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00703                                                        "has not been initialised")
00704     ;
00705     }
00706     assert (_sevmgrServiceContext != NULL);
00707     const stdair::STDAIR_Service& lSTDAIR_Service =
00708 *_sevmgrServiceContext->getSTDAIR_Service();
00709
00710 //

```

```

00711     return lSTDAIR_Service;
00712 }
00713
00715 const stdair::ProgressStatus& SEVMGR_Service::getStatus
() const {
00716
00717     // Retrieve the SEvMgr service context
00718     if (_sevmgrServiceContext == NULL) {
00719         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00720                                                     "has not been initialised")
;
00721     }
00722     assert (_sevmgrServiceContext != NULL);
00723     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00724         *_sevmgrServiceContext;
00725
00726     // Retrieve the event queue object instance
00727     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00728
00729     // Delegate the call to the dedicated function
00730     return EventQueueManager::getStatus(lQueue);
00731
00732 }
00733
00735 const stdair::ProgressStatus& SEVMGR_Service::
00736 getStatus(const stdair::EventType::EN_EventType& iEventType) const
{
00737
00738     // Retrieve the SEvMgr service context
00739     if (_sevmgrServiceContext == NULL) {
00740         throw stdair::NonInitialisedServiceException ("The SEvMgr service "
00741                                                     "has not been initialised")
;
00742     }
00743     assert (_sevmgrServiceContext != NULL);
00744     SEVMGR_ServiceContext& lSEVMGR_ServiceContext =
00745         *_sevmgrServiceContext;
00746
00747     // Retrieve the event queue object instance
00748     EventQueue& lQueue = lSEVMGR_ServiceContext.getEventQueue();
00749
00750     // Delegate the call to the dedicated function
00751     return EventQueueManager::getStatus(lQueue,
iEventType);
00752
00753 }
00754
00755
00756 }

```

23.55 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](#)

23.56 SEVMGR_ServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```



```

00007 // StdAir
00008 #include <stdair/STDAIR_Service.hpp>
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/factory/FacBom.hpp>
00011 // SEvMgr
00012 #include <sevmgr/basic/BasConst_EventQueueManager.hpp>
00013 >
00014 #include <sevmgr/bom/EventQueue.hpp>
00015 #include <sevmgr/service/SEVMGR_ServiceContext.hpp>
00016 >
00015 namespace SEVMGR {
00016 // //////////////////////////////////////
00017 SEVMGR_ServiceContext::SEVMGR_ServiceContext()
00018 : _eventQueue (NULL) {
00019     init();
00020 }
00021 // //////////////////////////////////////
00022 SEVMGR_ServiceContext::
00023 SEVMGR_ServiceContext (const SEVMGR_ServiceContext& iServiceContext)
00024 : _eventQueue (iServiceContext._eventQueue) {
00025 }
00026 // //////////////////////////////////////
00027 SEVMGR_ServiceContext::~SEVMGR_ServiceContext() {
00028 }
00029 // //////////////////////////////////////
00030 void SEVMGR_ServiceContext::init() {
00031     //
00032     initEventQueue();
00033 }
00034 // //////////////////////////////////////
00035 void SEVMGR_ServiceContext::initEventQueue() {
00036     // The event queue key is just a string. For now, it is not used.
00037     const EventQueueKey lKey ("EQ01");
00038     // Create an EventQueue object instance
00039     EventQueue& lEventQueue = stdair::FacBom<EventQueue>::instance().create (
00040         lKey);
00041     // Store the event queue object
00042     _eventQueue = &lEventQueue;
00043 }
00044 // //////////////////////////////////////
00045 const std::string SEVMGR_ServiceContext::shortDisplay() const {
00046     std::ostringstream oStr;
00047     oStr << "SEVMGR_ServiceContext -- Owns StdAir service: "
00048         << _ownStdairService;
00049     if (_eventQueue != NULL) {
00050         oStr << " -- Queue: " << _eventQueue->toString();
00051     }
00052     return oStr.str();
00053 }
00054 // //////////////////////////////////////
00055 const std::string SEVMGR_ServiceContext::display() const {
00056     std::ostringstream oStr;
00057     oStr << shortDisplay();
00058     return oStr.str();
00059 }
00060 // //////////////////////////////////////
00061 const std::string SEVMGR_ServiceContext::describe() const {
00062     return shortDisplay();
00063 }
00064 // //////////////////////////////////////
00065 void SEVMGR_ServiceContext::reset() {
00066     // The shared_ptr<>::reset() method drops the refcount by one.
00067     // If the count result is dropping to zero, the resource pointed to
00068     // by the shared_ptr<> will be freed.
00069     // Reset the stdair shared pointer
00070     _stdairService.reset();
00071 }
00072 // //////////////////////////////////////
00073 EventQueue& SEVMGR_ServiceContext::getEventQueue() const {
00074     assert (_eventQueue != NULL);
00075     return *_eventQueue;
00076 }

```

```

00091     }
00092
00093 }

```

23.57 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](#)

23.58 SEVMGR_ServiceContext.hpp

```

00001 #ifndef __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP
00002 #define __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_service_types.hpp>
00012 #include <stdair/service/ServiceAbstract.hpp>
00013 // SEvMgr
00014 #include <sevmgr/SEVMGR_Types.hpp>
00015
00016 namespace stdair {
00017     class FacBomManager;
00018     template <typename BOM> class FacBom;
00019 }
00020
00021 namespace SEVMGR {
00022     class EventQueue;
00023
00024     class SEVMGR_ServiceContext : public
00025         stdair::ServiceAbstract {
00026     public:
00027         friend class SEVMGR_Service;
00028         friend class FacSEVMGRServiceContext;
00029
00030     private:
00031         // ////////////////////////////////////// Getters //////////////////////////////////////
00032         stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00033             return _stdairService;
00034         }
00035
00036         stdair::STDAIR_Service& getSTDAIR_Service() const {
00037             assert (_stdairService != NULL);
00038             return *_stdairService;
00039         }
00040
00041         const bool getOwnStdairServiceFlag() const {
00042             return _ownStdairService;
00043         }
00044     }
00045 }

```

```

00066     EventQueue& getEventQueue() const;
00067
00068
00069 private:
00070     // ////////// Setters //////////
00074     void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00075                             const bool iOwnStdairService) {
00076         _stdairService = ioSTDAIR_ServicePtr;
00077         _ownStdairService = iOwnStdairService;
00078     }
00079
00080
00081 private:
00082     // ////////// Display Methods //////////
00086     const std::string shortDisplay() const;
00087
00091     const std::string display() const;
00092
00096     const std::string describe() const;
00097
00098
00099 private:
00101     SEVMGR_ServiceContext();
00108     SEVMGR_ServiceContext (const SEVMGR_ServiceContext&);
00109
00113     ~SEVMGR_ServiceContext();
00114
00118     void reset();
00119
00127     void init();
00128
00135     void initEventQueue();
00136
00137
00138 private:
00139     // ////////////////////////////////// Children //////////////////////////////////
00143     stdair::STDAIR_ServicePtr_T _stdairService;
00144
00148     bool _ownStdairService;
00149
00150
00151 private:
00152     // ////////////////////////////////// Attributes //////////////////////////////////
00156     EventQueue* _eventQueue;
00157 };
00158
00159 }
00160 #endif // __SEVMGR_SVC_SEVMGRSERVICECONTEXT_HPP

```

23.59 sevmgr/SEVMGR_Exceptions.hpp File Reference

```

#include <exception>
#include <stdair/stdair_exceptions.hpp>

```

Classes

- class [SEVMGR::SEvMgrException](#)
- class [SEVMGR::EventQueueException](#)

Namespaces

- namespace [SEVMGR](#)

23.60 SEVMGR_Exceptions.hpp

```

00001 #ifndef __SEVMGR_SEVMGR_EXCEPTIONS_HPP
00002 #define __SEVMGR_SEVMGR_EXCEPTIONS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////

```

```

00007 // STL
00008 #include <exception>
00009 // StdAir
00010 #include <stdair/stdair_exceptions.hpp>
00011
00012 namespace SEVMGR {
00013
00014     // ////////// Exceptions //////////
00018     class SEvMgrException : public stdair::RootException {
00019     public:
00023         SEvMgrException (const std::string& iWhat)
00024             : stdair::RootException (iWhat) {}
00025     };
00026
00028     class EventQueueException : public SEvMgrException
00029     {
00029     public:
00031         EventQueueException (const std::string& iWhat) :
SEvMgrException (iWhat) {}
00032     };
00033
00034
00035 }
00036 #endif // __SEVMGR_SEVMGR_EXCEPTIONS_HPP
00037

```

23.61 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](#)

23.62 SEVMGR_Service.hpp

```

00001 #ifndef __SEVMGR_SEVMGR_SERVICE_HPP
00002 #define __SEVMGR_SEVMGR_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_json.hpp>
00010 #include <stdair/stdair_service_types.hpp>
00011 #include <stdair/bom/EventTypes.hpp>
00012 #include <stdair/bom/EventStruct.hpp>
00013
00014 // Forward declarations
00015 namespace stdair {
00016     struct ProgressStatusSet;
00017     struct BasLogParams;
00018     struct BasDBParams;
00019     struct BookingRequestStruct;
00020 }
00021
00022 namespace SEVMGR {

```

```

00023
00025     class SEVMGR_ServiceContext;
00026     class EventQueue;
00027     //struct EventStruct;
00028
00032     class SEVMGR_Service {
00033     public:
00034         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00050         SEVMGR_Service (const stdair::BasLogParams&, const
stdair::BasDBParams&);
00051
00063         SEVMGR_Service (const stdair::BasLogParams&);
00064
00080         SEVMGR_Service (stdair::STDAIR_ServicePtr_T);
00081
00085         ~SEVMGR_Service();
00086
00087
00088     public:
00089         // ////////////////////////////////// Business support methods //////////////////////////////////
00093         void buildSampleQueue();
00094
00125         stdair::BookingRequestStruct buildSampleBookingRequest
(const bool isForCRS = false);
00126
00143         stdair::ProgressStatusSet popEvent (stdair::EventStruct&) const;
00144
00145
00153         void run (stdair::EventStruct&) const;
00154
00169         bool select (stdair::EventStruct&,
const stdair::DateTime_T&) const;
00170
00171
00180         template<class EventGenerator>
00181         void addEventGenerator (EventGenerator& iEventGenerator)
const;
00182
00186         void addEvent (stdair::EventStruct&) const;
00187
00192         void reset() const;
00193
00203         void updateStatus (const stdair::EventType::EN_EventType&,
const stdair::Count_T&) const;
00204
00205
00215         void addStatus (const stdair::EventType::EN_EventType&,
const stdair::Count_T&) const;
00216
00217
00223         bool isQueueDone() const;
00224
00229         bool hasProgressStatus(const
stdair::EventType::EN_EventType&) const;
00230
00231         /* @brief Get a reference on the EventQueue object.
00232          *
00233          * @return EventQueue& Reference on the EventQueue.
00234          */
00235         EventQueue& getEventQueue() const;
00236
00240         const stdair::Count_T& getQueueSize() const;
00241
00251         template<class EventGenerator, class Key>
00252         EventGenerator& getEventGenerator(const Key& iKey) const;
00253
00263         template<class EventGenerator, class Key>
00264         bool hasEventGenerator(const Key& iKey) const;
00265
00275         template<class EventGenerator>
00276         const std::list<EventGenerator*> getEventGeneratorList
() const;
00277
00287         template<class EventGenerator>
00288         bool hasEventGeneratorList() const;
00289
00304         const stdair::Count_T& getExpectedTotalNumberOfEventsToBeGenerated
() const;
00305
00322         const stdair::Count_T&
getExpectedTotalNumberOfEventsToBeGenerated
(const stdair::EventType::EN_EventType&) const;
00323
00324
00337         const stdair::Count_T& getActualTotalNumberOfEventsToBeGenerated
() const;
00338
00355         const stdair::Count_T&
getActualTotalNumberOfEventsToBeGenerated
(const stdair::EventType::EN_EventType&) const;
00356
00357

```

```

00361     const stdair::ProgressStatus& getStatus () const;
00362
00367     const stdair::ProgressStatus& getStatus (const
stdair::EventType::EN_EventType&) const;
00368
00369 public:
00370     // //////////////// Display support methods ////////////////
00371
00378     std::string describeKey() const;
00379
00386     std::string list () const;
00387
00397     std::string list (const stdair::EventType::EN_EventType&) const;
00398
00399 public:
00400     // //////////////// Export support methods ////////////////
00409     std::string jsonHandler (const stdair::JSONString&) const;
00410
00415     std::string jsonExportEventQueue (const
stdair::EventType::EN_EventType& =
00416                                     stdair::EventType::LAST_VALUE) const;
00417
00421     std::string jsonExportEvent (const stdair::EventStruct&
const;
00422
00423 private:
00424     // //////////////// Constructors and Destructors ////////////////
00428     SEVMGR_Service();
00429
00433     SEVMGR_Service (const SEVMGR_Service&);
00434
00439     void initServiceContext();
00440
00452     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
const stdair::BasDBParams&);
00453
00454     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
;
00465
00474     void addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
const bool iOwnStdairService);
00475
00476
00483     void initSevmgrService();
00484
00488     void finalise();
00489
00490 private:
00491     // //////////////// Getters ////////////////
00498     const stdair::STDAIR_Service& getSTDAIR_Service() const;
00499
00500 private:
00501     // //////////////// Service Context ////////////////
00505     SEVMGR_ServiceContext* _sevmgrServiceContext;
00506 };
00507
00508 }
00509 #endif // __SEVMGR_SEVMGR_SERVICE_HPP

```

23.63 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

23.64 SEVMGR_Types.hpp

```

00001 #ifndef __SEVMGR_SEVMGR_TYPES_HPP
00002 #define __SEVMGR_SEVMGR_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // Boost
00008 #include <boost/shared_ptr.hpp>
00009 // Stdair
00010 #include <stdair/basic/ProgressStatusSet.hpp>
00011 #include <stdair/basic/EventType.hpp>
00012 // Sevmgr
00013 #include <sevmgr/SEVMGR_Exceptions.hpp>
00014
00015 namespace SEVMGR {
00016
00017     // Forward declarations
00018     class SEVMGR_Service;
00019
00020     // ////////// Type definitions specific to to Sevmgr //////////
00024     typedef boost::shared_ptr<SEVMGR_Service> SEVMGR_ServicePtr_T
00025 ;
00027     typedef std::string EventQueueID_T;
00028
00034     typedef std::map<stdair::EventType::EN_EventType,
00035                     stdair::ProgressStatus> ProgressStatusMap_T
00036 ;
00037 }
00038 #endif // __SEVMGR_SEVMGR_TYPES_HPP
00039

```

23.65 sevmgr/ui/cmdline/sevmgr.cpp File Reference

23.66 sevmgr.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 #include <boost/regex.hpp>
00015 #include <boost/swap.hpp>
00016 #include <boost/algorithm/string/case_conv.hpp>
00017 // StdAir
00018 #include <stdair/stdair_exceptions.hpp>
00019 #include <stdair/basic/BasLogParams.hpp>
00020 #include <stdair/basic/BasDBParams.hpp>
00021 #include <stdair/service/Logger.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/bom/BookingRequestTypes.hpp>
00024 #include <stdair/basic/ProgressStatusSet.hpp>
00025 #include <stdair/bom/EventStruct.hpp>
00026 // GNU Readline Wrapper
00027 #include <stdair/ui/cmdline/SReadline.hpp>
00028 // SEvMgr
00029 #include <sevmgr/SEVMGR_Service.hpp>
00030 #include <sevmgr/config/sevmgr-paths.hpp>
00031
00032 // ////////// Constants //////////
00036 const std::string K_SEVMGR_DEFAULT_LOG_FILENAME ("
    sevmgr.log");
00037
00041 const int K_SEVMGR_EARLY_RETURN_STATUS = 99;
00042
00047 typedef std::vector<std::string> TokenList_T;
00048

```

```

00052 struct Command_T {
00053     typedef enum {
00054         NOP = 0,
00055         QUIT,
00056         HELP,
00057         LIST,
00058         DISPLAY,
00059         SELECT,
00060         NEXT,
00061         RUN,
00062         JSON_LIST,
00063         JSON_DISPLAY,
00064         LAST_VALUE
00065     } Type_T;
00066 };
00067
00068 // ////////// Parsing of Options & Configuration //////////
00069 // A helper function to simplify the main part.
00070 template<class T> std::ostream& operator<< (std::ostream& os,
00071     const std::vector<T>& v) {
00072     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00073     return os;
00074 }
00075
00076 int readConfiguration (int argc, char* argv[], std::string&
ioLogFilename) {
00077     // Declare a group of options that will be allowed only on command line
00078     boost::program_options::options_description generic ("Generic options");
00079     generic.add_options()
00080         ("prefix", "print installation prefix")
00081         ("version,v", "print version string")
00082         ("help,h", "produce help message");
00083
00084     // Declare a group of options that will be allowed both on command
00085     // line and in config file
00086
00087     boost::program_options::options_description config ("Configuration");
00088     config.add_options()
00089         ("log,l",
00090         boost::program_options::value< std::string >(&ioLogFilename)->
00091         default_value(K_SEVMGR_DEFAULT_LOG_FILENAME),
00092         "Filename for the logs")
00093         ;
00094
00095     // Hidden options, will be allowed both on command line and
00096     // in config file, but will not be shown to the user.
00097     boost::program_options::options_description hidden ("Hidden options");
00098     hidden.add_options()
00099         ("copyright",
00100         boost::program_options::value< std::vector<std::string> >(),
00101         "Show the copyright (license)");
00102
00103     boost::program_options::options_description cmdline_options;
00104     cmdline_options.add(generic).add(config).add(hidden);
00105
00106     boost::program_options::options_description config_file_options;
00107     config_file_options.add(config).add(hidden);
00108     boost::program_options::options_description visible ("Allowed options");
00109     visible.add(generic).add(config);
00110
00111     boost::program_options::positional_options_description p;
00112     p.add ("copyright", -1);
00113
00114     boost::program_options::variables_map vm;
00115     boost::program_options::store (boost::program_options::command_line_parser (argc, argv).
00116     options (cmdline_options).positional(p).run(), vm);
00117
00118     std::ifstream ifs ("sevmgr.cfg");
00119     boost::program_options::store (parse_config_file (ifs, config_file_options),
00120     vm);
00121     boost::program_options::notify (vm);
00122
00123     if (vm.count ("help")) {
00124         std::cout << visible << std::endl;
00125         return K_SEVMGR_EARLY_RETURN_STATUS;
00126     }
00127
00128     if (vm.count ("version")) {
00129         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
00130         << std::endl;
00131         return K_SEVMGR_EARLY_RETURN_STATUS;
00132     }
00133
00134     if (vm.count ("prefix")) {
00135         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00136         return K_SEVMGR_EARLY_RETURN_STATUS;
00137     }
00138

```



```

00139     }
00140
00141     if (vm.count ("log")) {
00142         ioLogFilename = vm["log"].as< std::string >();
00143         std::cout << "Log filename is: " << ioLogFilename << std::endl;
00144     }
00145
00146     return 0;
00147 }
00148
00149 ///////////////////////////////////////////////////////////////////
00150 void initReadline (swift::SReadline& ioInputReader) {
00151
00152     // Prepare the list of my own completers
00153     std::vector<std::string> Completers;
00154
00155     // The following is supported:
00156     // - "identifiers"
00157     // - special identifier %file - means to perform a file name completion
00158     Completers.push_back ("help");
00159     Completers.push_back ("list");
00160     Completers.push_back ("list BookingRequest");
00161     Completers.push_back ("list BreakPoint");
00162     Completers.push_back ("select %date %time");
00163     Completers.push_back ("display");
00164     Completers.push_back ("next");
00165     Completers.push_back ("run");
00166     Completers.push_back ("json_list");
00167     Completers.push_back ("json_display");
00168     Completers.push_back ("quit");
00169
00170
00171     // Now register the completers.
00172     // Actually it is possible to re-register another set at any time
00173     ioInputReader.RegisterCompletions (Completers);
00174 }
00175
00176 ///////////////////////////////////////////////////////////////////
00177 void parseEventDateTime (const TokenList_T& iTokenList,
00178                         stdair::Date_T& ioEventData,
00179                         stdair::Duration_T& ioEventTime) {
00180     //
00181     const std::string kMonthStr[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
00182                                         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
00183     //
00184     unsigned short ioEventDataYear = ioEventData.year();
00185     unsigned short ioEventDataMonth = ioEventData.month();
00186     std::string ioEventDataMonthStr = kMonthStr[ioEventDataMonth-1];
00187     unsigned short ioEventDataDay = ioEventData.day();
00188     //
00189     unsigned short ioEventTimeHours = ioEventTime.hours();
00190     unsigned short ioEventTimeMinutes = ioEventTime.minutes();
00191     unsigned short ioEventTimeSeconds = ioEventTime.seconds();
00192
00193     // Interpret the user input
00194     if (iTokenList.empty() == false) {
00195
00196         // Read the date year
00197         TokenList_T::const_iterator itTok = iTokenList.begin();
00198
00199         // Read the year for the event date
00200         if (itTok != iTokenList.end()) {
00201
00202             if (itTok->empty() == false) {
00203                 try {
00204
00205                     ioEventDataYear = boost::lexical_cast<unsigned short> (*itTok);
00206                     if (ioEventDataYear < 100) {
00207                         ioEventDataYear += 2000;
00208                     }
00209
00210                 } catch (boost::bad_lexical_cast& eCast) {
00211                     std::cerr << "The year of the event date (" << *itTok
00212                                 << ") cannot be understood. The default value ("
00213                                 << ioEventDataYear << ") is kept. " << std::endl;
00214                     return;
00215                 }
00216             }
00217
00218         } else {
00219             return;
00220         }
00221
00222         // Read the month for the event date
00223         ++itTok;
00224         if (itTok != iTokenList.end()) {
00225

```

```

00226     if (itTok->empty() == false) {
00227         try {
00228
00229             const boost::regex lMonthRegex ("^\\d{1,2}$");
00230             const bool isMonthANumber = regex_match (*itTok, lMonthRegex);
00231
00232             if (isMonthANumber == true) {
00233                 const unsigned short lMonth =
00234                     boost::lexical_cast<unsigned short> (*itTok);
00235                 if (lMonth > 12) {
00236                     throw boost::bad_lexical_cast();
00237                 }
00238                 ioEventDataMonthStr = kMonthStr[lMonth-1];
00239             }
00240         } else {
00241             const std::string lMonthStr (*itTok);
00242             if (lMonthStr.size() < 3) {
00243                 throw boost::bad_lexical_cast();
00244             }
00245             std::string lMonthStr1 (lMonthStr.substr (0, 1));
00246             boost::algorithm::to_upper (lMonthStr1);
00247             std::string lMonthStr23 (lMonthStr.substr (1, 2));
00248             boost::algorithm::to_lower (lMonthStr23);
00249             ioEventDataMonthStr = lMonthStr1 + lMonthStr23;
00250         }
00251     }
00252     } catch (boost::bad_lexical_cast& eCast) {
00253         std::cerr << "The month of the event date ('" << *itTok
00254             << "') cannot be understood. The default value ("
00255             << ioEventDataMonthStr << ") is kept. " << std::endl;
00256         return;
00257     }
00258 }
00259
00260 // Read the day for the event date
00261 ++itTok;
00262 if (itTok != iTokenList.end()) {
00263     if (itTok->empty() == false) {
00264         try {
00265             ioEventDataDay = boost::lexical_cast<unsigned short> (*itTok);
00266         } catch (boost::bad_lexical_cast& eCast) {
00267             std::cerr << "The day of the event date ('" << *itTok
00268                 << "') cannot be understood. The default value ("
00269                 << ioEventDataDay << ") is kept. " << std::endl;
00270             return;
00271         }
00272     }
00273 } else {
00274     return;
00275 }
00276
00277 // Re-compose the event date
00278 std::ostringstream lEventDateStr;
00279 lEventDateStr << ioEventDataYear << "-" << ioEventDataMonthStr
00280     << "-" << ioEventDataDay;
00281
00282 try {
00283     ioEventData =
00284         boost::gregorian::from_simple_string (lEventDateStr.str());
00285 } catch (boost::gregorian::bad_month& eCast) {
00286     std::cerr << "The event date ('" << lEventDateStr.str()
00287         << "') cannot be understood. The default value ("
00288         << ioEventData << ") is kept. " << std::endl;
00289     return;
00290 }
00291
00292 // Read the hours of the event time
00293 ++itTok;
00294 if (itTok != iTokenList.end()) {
00295     if (itTok->empty() == false) {
00296         try {
00297             ioEventTimeHours = boost::lexical_cast<unsigned short> (*itTok);
00298         } catch (boost::bad_lexical_cast& eCast) {
00299             std::cerr << "The hours of the event time ('" << *itTok
00300                 << "') cannot be understood. The default value ("
00301                 << ioEventTimeHours << ") is kept. " << std::endl;
00302             return;
00303         }
00304     }
00305 }

```

```

00313     }
00314 }
00315
00316 } else {
00317     return;
00318 }
00319
00320 // Read the minutes of the event time
00321 ++itTok;
00322 if (itTok != iTokenList.end()) {
00323     if (itTok->empty() == false) {
00324         try {
00325             ioEventTimeMinutes = boost::lexical_cast<unsigned short> (*itTok);
00326         } catch (boost::bad_lexical_cast& eCast) {
00327             std::cerr << "The minutes of the event time ('" << *itTok
00328                 << "') cannot be understood. The default value ("
00329                 << ioEventTimeMinutes << ") is kept. " << std::endl;
00330             return;
00331         }
00332     }
00333 } else {
00334     return;
00335 }
00336
00337 // Read the seconds of the event time
00338 ++itTok;
00339 if (itTok != iTokenList.end()) {
00340     if (itTok->empty() == false) {
00341         try {
00342             ioEventTimeSeconds = boost::lexical_cast<unsigned short> (*itTok);
00343         } catch (boost::bad_lexical_cast& eCast) {
00344             std::cerr << "The seconds of the event time ('" << *itTok
00345                 << "') cannot be understood. The default value ("
00346                 << ioEventTimeSeconds << ") is kept. " << std::endl;
00347             return;
00348         }
00349     }
00350 } else {
00351     return;
00352 }
00353
00354 // Re-compose the event time
00355 std::ostringstream lEventTimeStr;
00356 lEventTimeStr << ioEventTimeHours << ":" << ioEventTimeMinutes
00357     << ":" << ioEventTimeSeconds;
00358
00359 try {
00360     ioEventTime =
00361         boost::posix_time::duration_from_string (lEventTimeStr.str());
00362 } catch (boost::gregorian::bad_month& eCast) {
00363     std::cerr << "The event time ('" << lEventTimeStr.str()
00364         << "') cannot be understood. The default value ("
00365         << ioEventTime << ") is kept. " << std::endl;
00366     return;
00367 }
00368
00369 }
00370
00371 }
00372
00373 // //////////////////////////////////////
00374 Command_T::Type_T extractCommand (TokenList_T& ioTokenList) {
00375     Command_T::Type_T oCommandType = Command_T::LAST_VALUE;
00376
00377     // Interpret the user input
00378     if (ioTokenList.empty() == false) {
00379         TokenList_T::iterator itTok = ioTokenList.begin();
00380         std::string lCommand (*itTok);
00381         boost::algorithm::to_lower (lCommand);
00382
00383         if (lCommand == "help") {
00384             oCommandType = Command_T::HELP;
00385         } else if (lCommand == "list") {
00386             oCommandType = Command_T::LIST;
00387         } else if (lCommand == "display") {

```

```

00400         oCommandType = Command_T::DISPLAY;
00401
00402     } else if (lCommand == "select") {
00403         oCommandType = Command_T::SELECT;
00404
00405     } else if (lCommand == "next") {
00406         oCommandType = Command_T::NEXT;
00407
00408     } else if (lCommand == "run") {
00409         oCommandType = Command_T::RUN;
00410
00411     } else if (lCommand == "json_list") {
00412         oCommandType = Command_T::JSON_LIST;
00413
00414     } else if (lCommand == "json_display") {
00415         oCommandType = Command_T::JSON_DISPLAY;
00416
00417     } else if (lCommand == "quit") {
00418         oCommandType = Command_T::QUIT;
00419     }
00420
00421     // Remove the first token (the command), as the corresponding information
00422     // has been extracted in the form of the returned command type enumeration
00423     ioTokenList.erase (itTok);
00424
00425     } else {
00426         oCommandType = Command_T::NOP;
00427     }
00428
00429     return oCommandType;
00430 }
00431
00432 // //////////////////////////////////////
00433 std::string toString (const TokenList_T& iTokenList) {
00434     std::ostringstream oStr;
00435
00436     // Re-create the string with all the tokens, trimmed by read-line
00437     unsigned short idx = 0;
00438     for (TokenList_T::const_iterator itTok = iTokenList.begin();
00439          itTok != iTokenList.end(); ++itTok, ++idx) {
00440         if (idx != 0) {
00441             oStr << " ";
00442         }
00443         oStr << *itTok;
00444     }
00445
00446     return oStr.str();
00447 }
00448
00449 // //////////////////////////////////////
00450 TokenList_T extractTokenList (const TokenList_T& iTokenList,
00451                               const std::string& iRegularExpression) {
00452     TokenList_T oTokenList;
00453
00454     // Re-create the string with all the tokens (which had been trimmed
00455     // by read-line)
00456     const std::string lFullLine = toString (iTokenList);
00457
00458     // See the caller for the regular expression
00459     boost::regex expression (iRegularExpression);
00460
00461     std::string::const_iterator start = lFullLine.begin();
00462     std::string::const_iterator end = lFullLine.end();
00463
00464     boost::match_results<std::string::const_iterator> what;
00465     boost::match_flag_type flags = boost::match_default | boost::format_sed;
00466     regex_search (start, end, what, expression, flags);
00467
00468     // Put the matched strings in the list of tokens to be returned back
00469     // to the caller
00470     const unsigned short lMatchSetSize = what.size();
00471     for (unsigned short matchIdx = 1; matchIdx != lMatchSetSize; ++matchIdx) {
00472         const std::string lMatchedString (std::string (what[matchIdx].first,
00473                                                         what[matchIdx].second));
00474         //if (lMatchedString.empty() == false) {
00475         oTokenList.push_back (lMatchedString);
00476         //}
00477     }
00478
00479     // DEBUG
00480     // std::cout << "After (token list): " << oTokenList << std::endl;
00481
00482     return oTokenList;
00483 }
00484
00485 // //////////////////////////////////////
00486 TokenList_T extractTokenListForDateTime (const TokenList_T& iTokenList) {

```

```

00499     const std::string lRegex("^([[:digit:]]{2,4})?[/-]?[[:space:]]*"
00500         "([[:alpha:]]{3}|[[:digit:]]{1,2})?[/-]?[[:space:]]*"
00501         "([[:digit:]]{1,2})?[[:space:]]*"
00502         "([[:digit:]]{1,2})?[:-]?[[:space:]]*"
00503         "([[:alpha:]]{3}|[[:digit:]]{1,2})?[:-]?[[:space:]]*"
00504         "([[:digit:]]{1,2})?[[:space:]]*$");
00505
00506     //
00507     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00508     return oTokenList;
00509 }
00510
00511 // ////////// M A I N //////////
00512 int main (int argc, char* argv[]) {
00513
00514     // Readline history
00515     const unsigned int lHistorySize (100);
00516     const std::string lHistoryFilename ("sevmgr.hist");
00517     const std::string lHistoryBackupFilename ("sevmgr.hist.bak");
00518
00519     // Default parameters for the interactive session
00520     stdair::EventStruct lCurrentInteractiveEventStruct;
00521     stdair::DateTime_T lCurrentInteractiveDateTime;
00522     stdair::EventType::EN_EventType lCurrentInteractiveEventType;
00523
00524     // Output log File
00525     stdair::Filename_T lLogFilename;
00526
00527     // Call the command-line option parser
00528     const int lOptionParserStatus = readConfiguration (argc,
00529         argv, lLogFilename);
00530
00531     if (lOptionParserStatus == K_SEVMGR_EARLY_RETURN_STATUS
00532     ) {
00533         return 0;
00534     }
00535
00536     // Set the log parameters
00537     std::ofstream logOutputFile;
00538     // Open and clean the log outputfile
00539     logOutputFile.open (lLogFilename.c_str());
00540     logOutputFile.clear();
00541
00542     // Initialise the inventory service
00543     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00544     SEVMGR::SEVMGR_Service sevmgrService (lLogParams);
00545
00546     // Build the sample event queue.
00547     sevmgrService.buildSampleQueue();
00548
00549     // Pop out the first event from the queue.
00550     sevmgrService.popEvent(lCurrentInteractiveEventStruct);
00551
00552     // DEBUG
00553     STDAIR_LOG_DEBUG ("=====");
00554     STDAIR_LOG_DEBUG ("=          Beginning of the interactive session          =");
00555     STDAIR_LOG_DEBUG ("=====");
00556
00557     // Initialise the GNU readline wrapper
00558     swift::SReadline lReader (lHistoryFilename, lHistorySize);
00559     initReadline (lReader);
00560
00561     // Now we can ask user for a line
00562     std::string lUserInput;
00563     bool EndOfInput (false);
00564     Command_T::Type_T lCommandType (Command_T::NOP);
00565
00566     while (lCommandType != Command_T::QUIT && EndOfInput == false) {
00567         // Update the interactive parameters which have not been updated yet
00568         lCurrentInteractiveDateTime = lCurrentInteractiveEventStruct.getEventTime (
00569             );
00570         lCurrentInteractiveEventType = lCurrentInteractiveEventStruct.getEventType (
00571             );
00572
00573         // Prompt
00574         std::ostringstream oPromptStr;
00575         oPromptStr << "sevmgr "
00576             << stdair::EventType::getTypeLabelAsString(
00577                 lCurrentInteractiveEventType)
00578             << " / " << lCurrentInteractiveDateTime << "> ";
00579
00580         // Call read-line, which will fill the list of tokens
00581         TokenList_T lTokenListByReadline;
00582         lUserInput = lReader.GetLine (oPromptStr.str(), lTokenListByReadline,
00583             EndOfInput);

```

```

00579
00580 // The history can be saved to an arbitrary file at any time
00581 lReader.SaveHistory (lHistoryBackupFilename);
00582
00583 // The end-of-input typically corresponds to a CTRL-D typed by the user
00584 if (EndOfInput) {
00585     std::cout << std::endl;
00586     break;
00587 }
00588
00589 // Interpret the user input
00590 lCommandType = extractCommand (lTokenListByReadline);
00591
00592 switch (lCommandType) {
00593
00594     // ////////////////////////////////// Help //////////////////////////////////
00595 case Command_T::HELP: {
00596     std::cout << std::endl;
00597     std::cout << "Commands: " << std::endl;
00598     std::cout << " help" << "\t\t" << "Display this help" << std::endl;
00599     std::cout << " quit" << "\t\t" << "Quit the application" << std::endl;
00600     std::cout << " list" << "\t\t" << "List events in the queue. It is "
00601         << "possible to filter events according to their types"
00602         << std::endl
00603         << "\t\t\t\t\t 'list_event BookingRequest' "
00604         << "list all the booking requests" << std::endl
00605         << "\t\t\t\t\t 'list_event BreakPoint' "
00606         << "list all the break points" << std::endl;
00607     std::cout << " select" << "\t\t"
00608         << "Select an event into the 'list' to become the current one.
For instance, try the command:\n"
00609         << "\t\t\t\t\t 'select 2011-May-14 00:00:00'"
00610         << std::endl;
00611     std::cout << " display" << "\t"
00612         << "Display the current event" << std::endl;
00613     std::cout << " next" << "\t\t"
00614         << "Play the current event and pop the next one from the queue"
00615         << std::endl;
00616     std::cout << " run" << "\t\t"
00617         << "Play all the events until the next break-point, if any"
00618         << std::endl;
00619     std::cout << " \nDebug Commands" << std::endl;
00620     std::cout << " json_list" << "\t"
00621         << "List events in the queue in a JSON format"
00622         << std::endl;
00623     std::cout << " json_display" << "\t"
00624         << "Display the current event in a JSON format"
00625         << std::endl;
00626     std::cout << std::endl;
00627     break;
00628 }
00629
00630 // ////////////////////////////////// Quit //////////////////////////////////
00631 case Command_T::QUIT: {
00632     break;
00633 }
00634
00635 // ////////////////////////////////// List //////////////////////////////////
00636 case Command_T::LIST: {
00637
00638     //
00639     std::ostringstream oEventListStr;
00640
00641     if (lTokenListByReadline.empty() == true) {
00642
00643         // If no parameter is given, list all the events in the queue
00644         oEventListStr << sevmgrService.list ();
00645
00646     } else if (lTokenListByReadline.size() == 1) {
00647
00648         assert (lTokenListByReadline.empty() == false);
00649         const std::string lEventTypeStr (lTokenListByReadline[0]);
00650
00651         // If exactly one parameter is given, try to convert it into
00652         // an event type
00653         try {
00654
00655             const stdair::EventType lEventType (lEventTypeStr);
00656             const stdair::EventType::EN_EventType& lActualEventType =
00657                 lEventType.getType();
00658             oEventListStr << sevmgrService.list (lActualEventType);
00659
00660         } catch (stdair::CodeConversionException e) {
00661             oEventListStr << "The event type '" << lEventTypeStr
00662                 << "' is not known. Try 'help' for "
00663                 << "more information on the 'list_event' command."
00664                 << std::endl;

```

```

00665     }
00666   } else {
00667
00668     // If more than one parameter is given, display an error message
00669     oEventListStr << "The event type is not understood: try 'help' for "
00670     << "more information on the 'list_event' command."
00671     << std::endl;
00672   }
00673   std::cout << oEventListStr.str() << std::endl;
00674   STDAIR_LOG_DEBUG (oEventListStr.str());
00675
00676   //
00677   break;
00678 }
00679
00680 // ////////////////////////////////// Select //////////////////////////////////
00681 case Command_T::SELECT: {
00682
00683   //
00684   TokenList_T lTokenList = extractTokenListForDateTime (
lTokenListByReadline);
00685   stdair::Date_T lUserData = lCurrentInteractiveDateTime.date();
00686   stdair::Duration_T lUserTime = lCurrentInteractiveDateTime.time_of_day();
00687   parseEventDateTime (lTokenList, lUserData, lUserTime);
00688
00689   std::cout << "Try to select event: "
00690   << lUserData << " " << lUserTime
00691   << std::endl;
00692
00693   const stdair::DateTime_T lUserDateTime =
00694     boost::posix_time::ptime (lUserData, lUserTime);
00695
00696   const bool hasSelectBeenSuccessful =
00697     sevmgrService.select (lCurrentInteractiveEventStruct,
00698     lUserDateTime);
00699
00700   std::cout << "Selection successful: "
00701   << hasSelectBeenSuccessful << std::endl;
00702
00703   //
00704   break;
00705 }
00706
00707 // ////////////////////////////////// Display //////////////////////////////////
00708 case Command_T::DISPLAY: {
00709   //
00710   std::cout << "Display" << std::endl;
00711
00712   // DEBUG
00713   std::ostream oEventStr;
00714   oEventStr << lCurrentInteractiveEventStruct.describe();
00715   std::cout << oEventStr.str() << std::endl;
00716   STDAIR_LOG_DEBUG (oEventStr.str());
00717
00718   //
00719   break;
00720 }
00721
00722 // ////////////////////////////////// Next //////////////////////////////////
00723 case Command_T::NEXT: {
00724   //
00725   std::cout << "Next" << std::endl;
00726
00727   if (sevmgrService.isQueueDone() == true) {
00728
00729     // DEBUG
00730     std::ostream oEmptyQueueStr;
00731     oEmptyQueueStr << "The event queue is empty: no event can be popped
out.";
00732     std::cout << oEmptyQueueStr.str() << std::endl;
00733     STDAIR_LOG_DEBUG (oEmptyQueueStr.str());
00734
00735     //
00736     break;
00737   }
00738
00739   // Get the next event from the event queue
00740   stdair::ProgressStatusSet lPPS =
00741     sevmgrService.popEvent (lCurrentInteractiveEventStruct);
00742
00743   // DEBUG
00744   std::ostream oEventStr;
00745   oEventStr << "Popped event: '"
00746   << lCurrentInteractiveEventStruct.describe() << "'.";
00747   std::cout << oEventStr.str() << std::endl;
00748   STDAIR_LOG_DEBUG (oEventStr.str());
00749

```

```

00750
00751     //
00752     break;
00753 }
00754
00755     // /////////////////////////////////// Run ///////////////////////////////////
00756     case Command_T::RUN: {
00757         //
00758         std::cout << "Run" << std::endl;
00759
00760         // Delegate the call to the dedicated service
00761         sevmgrService.run (lCurrentInteractiveEventStruct);
00762         lCurrentInteractiveEventType = lCurrentInteractiveEventStruct.
getEventType ();
00763
00764         // DEBUG
00765         if (lCurrentInteractiveEventType == stdair::EventType::BRK_PT) {
00766             std::ostringstream oBreakPointStr;
00767             oBreakPointStr << "Break point found. Stop at: "
00768                 << lCurrentInteractiveEventStruct.describe() << ".";
00769             std::cout << oBreakPointStr.str() << std::endl;
00770             STDAIR_LOG_DEBUG (oBreakPointStr.str());
00771         } else {
00772             std::ostringstream oNoBreakPointStr;
00773             oNoBreakPointStr << "No break point found. All the events have been
played.\n"
00774                 << "The current event is the last one.";
00775             std::cout << oNoBreakPointStr.str() << std::endl;
00776             STDAIR_LOG_DEBUG (oNoBreakPointStr.str());
00777         }
00778
00779         //
00780         break;
00781     }
00782
00783     // /////////////////////////////////// JSon List ///////////////////////////////////
00784
00785     case Command_T::JSON_LIST: {
00786         //
00787         std::cout << "JSON List" << std::endl;
00788
00789         // Delegate the call to the dedicated service
00790         const std::string& lCSVEventQueueDumpAfter =
sevmgrService.jsonExportEventQueue ();
00791
00792         // DEBUG: Display the events queue JSON string
00793         std::cout << lCSVEventQueueDumpAfter << std::endl;
00794         STDAIR_LOG_DEBUG (lCSVEventQueueDumpAfter);
00795
00796         break;
00797     }
00798
00799     // /////////////////////////////////// JSon Display ///////////////////////////////////
00800
00801     case Command_T::JSON_DISPLAY: {
00802         //
00803         std::cout << "JSON Display" << std::endl;
00804
00805         // Delegate the call to the dedicated service
00806         const std::string& lCSVEventDumpAfter =
sevmgrService.jsonExportEvent (lCurrentInteractiveEventStruct);
00807
00808         // DEBUG: Display the event JSON string
00809         std::cout << lCSVEventDumpAfter << std::endl;
00810         STDAIR_LOG_DEBUG (lCSVEventDumpAfter);
00811
00812         //
00813         break;
00814     }
00815
00816     // /////////////////////////////////// Default / No value ///////////////////////////////////
00817
00818     case Command_T::NOP: {
00819         break;
00820     }
00821
00822     case Command_T::LAST_VALUE:
00823     default: {
00824         // DEBUG
00825         std::ostringstream oStr;
00826         oStr << "That command is not yet understood: " << lUserInput
00827             << " => " << lTokenListByReadline;
00828         STDAIR_LOG_DEBUG (oStr.str());
00829         std::cout << oStr.str() << std::endl;
00830     }
00831 }
00832 }
00833 }
00834

```



```

00835 // DEBUG
00836 STDAIR_LOG_DEBUG ("End of the session. Exiting.");
00837 std::cout << "End of the session. Exiting." << std::endl;
00838
00839 // Close the Log outputFile
00840 logOutputFile.close();
00841
00842 /*
00843 Note: as that program is not intended to be run on a server in
00844 production, it is better not to catch the exceptions. When it
00845 happens (that an exception is throwned), that way we get the
00846 call stack.
00847 */
00848
00849 return 0;
00850 }

```

23.67 test/sevmgr/EventQueueManagementTestSuite.cpp File Reference

23.68 EventQueueManagementTestSuite.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <map>
00012 #include <cmath>
00013 // Boost Unit Test Framework (UTF)
00014 #define BOOST_TEST_DYN_LINK
00015 #define BOOST_TEST_MAIN
00016 #define BOOST_TEST_MODULE EventQueueManagementTest
00017 #include <boost/test/unit_test.hpp>
00018 #include <boost/shared_ptr.hpp>
00019 // StdAir
00020 #include <stdair/stdair_basic_types.hpp>
00021 #include <stdair/stdair_date_time_types.hpp>
00022 #include <stdair/basic/BasLogParams.hpp>
00023 #include <stdair/basic/BasDBParams.hpp>
00024 #include <stdair/basic/BasFileMgr.hpp>
00025 #include <stdair/basic/ProgressStatusSet.hpp>
00026 #include <stdair/bom/EventStruct.hpp>
00027 #include <stdair/bom/BookingRequestStruct.hpp>
00028 #include <stdair/bom/BookingRequestTypes.hpp>
00029 #include <stdair/service/Logger.hpp>
00030 // SEvMgr
00031 #include <sevmgr/SEVMGR_Service.hpp>
00032 #include <sevmgr/config/sevmgr-paths.hpp>
00033
00034 namespace boost_utf = boost::unit_test;
00035
00036 // (Boost) Unit Test XML Report
00037 std::ofstream utfReportStream ("EventQueueManagementTestSuite_utfresults.xml");
00038
00042 struct UnitTestConfig {
00044     UnitTestConfig() {
00045         boost_utf::unit_test_log.set_stream (utfReportStream);
00046         boost_utf::unit_test_log.set_format (boost_utf::XML);
00047         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00048         //boost_utf::unit_test_log.set_threshold_level
00049         (boost_utf::log_successful_tests);
00050     }
00052     ~UnitTestConfig() {
00053     }
00054 };
00055
00056 // Specific type definitions
00057 typedef std::pair<stdair::Count_T, stdair::Count_T> NbOfEventsPair_T;
00058 typedef std::map<const stdair::DemandStreamKeyStr_T,
00059                 NbOfEventsPair_T> NbOfEventsByDemandStreamMap_T;
00060
00061
00062 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00063
00064 // Set the UTF configuration (re-direct the output to a specific file)
00065 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00066
00067 // Start the test suite
00068 BOOST_AUTO_TEST_SUITE (master_test_suite)
00069

```

```

00070
00073 BOOST_AUTO_TEST_CASE (sevmgr_simple_simulation_test) {
00074
00075     // Output log File
00076     const stdair::Filename_T lLogFilename ("EventQueueManagementTestSuite.log");
00077
00078     // Set the log parameters
00079     std::ofstream logOutputFile;
00080     // open and clean the log outputfile
00081     logOutputFile.open (lLogFilename.C_str());
00082     logOutputFile.clear();
00083
00084     // Initialise the Sevmgr service object
00085     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00086     SEVMGR::SEVMGR_Service sevmgrService (lLogParams);
00087
00088     const bool isQueueDone = sevmgrService.isQueueDone();
00089     BOOST_REQUIRE_MESSAGE (isQueueDone == true,
00090         "The event queue should be empty at this step. No "
00091         "<< \"insertion done.\"");
00092
00093     sevmgrService.buildSampleQueue ();
00094
00095     stdair::Count_T lNbOfEvents (sevmgrService.getQueueSize());
00096
00097     BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == false,
00098         "The event queue should not be empty at this step. "
00099         "<< \"Two insertions done.\"");
00100
00101     stdair::Count_T idx = 1;
00102     while (sevmgrService.isQueueDone() == false) {
00103
00104         // Pop the next event out of the event queue
00105         stdair::EventStruct lEventStruct;
00106         const stdair::ProgressStatusSet lPPS =
00107             sevmgrService.popEvent (lEventStruct);
00108
00109         // DEBUG
00110         STDAIR_LOG_DEBUG ("Poped event "<< idx << ": "
00111             << lEventStruct.describe() << "'.");
00112         STDAIR_LOG_DEBUG ("Progressss status: " << lPPS.describe());
00113         STDAIR_LOG_DEBUG ("Poped event: "
00114             << lEventStruct.describe() << "'.");
00115
00116         // Iterate
00117         ++idx;
00118     }
00119
00120     // Compensate for the last iteration
00121     --idx;
00122     // Compared the actual number of popped events with the expected one.
00123     BOOST_REQUIRE_MESSAGE (idx == lNbOfEvents,
00124         "Actual number of requests in the queue: "
00125         "<< idx << ". Expected value: " << lNbOfEvents);
00126
00127     BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
00128         "The event queue should be empty at this step: "
00129         "the two events have been popped.");
00130
00131     STDAIR_LOG_DEBUG ("Re-added the events into the queue");
00132
00133     // Add again the four events into the queue thanks to
00134     // sevmgrService.buildSampleQueue().
00135     // Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
00136     // Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
00137     sevmgrService.buildSampleQueue ();
00138
00139     // Pop the next event out of the event queue
00140     stdair::EventStruct lFirstEventStruct;
00141     const stdair::ProgressStatusSet lFirstPS =
00142         sevmgrService.popEvent (lFirstEventStruct);
00143
00144     // Extract the corresponding date
00145     const stdair::DateTime_T& lFirstEventDateTime =
00146         lFirstEventStruct.getEventTime ();
00147     const stdair::Date_T& lFirstRequestDate =
00148         lFirstEventDateTime.date();
00149
00150     const stdair::Date_T lExpectedDate (2010, boost::gregorian::Jan, 21);
00151     BOOST_REQUIRE_MESSAGE (lFirstRequestDate == lExpectedDate,
00152         "Date of the first event popped from the queue: "
00153         "<< lFirstRequestDate << ". Should be: "
00154         "<< lExpectedDate << " which is earlier in time.");
00155
00156     STDAIR_LOG_DEBUG ("Reset the queue");
00157     sevmgrService.reset();
00158
00159

```

```

00178 BOOST_REQUIRE_MESSAGE (sevmgrService.isQueueDone() == true,
00179                          "The event queue has been reset: it should be empty "
00180                          "<< \"at this step.\");
00181
00182 STDAIR_LOG_DEBUG ("Re-added the events into the queue one more time");
00183
00184 // Add again the four events into the queue thanks to
00185 // sevmgrService.buildSampleQueue().
00186 // Dates of the break points: 21-JAN-2010 and 14-MAY-2011.
00187 // Dates of the booking requests: 22-JAN-2010 and 15-MAY-2011.
00188 sevmgrService.buildSampleQueue ();
00189
00192 stdair::EventStruct lBreakPointStruct;
00193 sevmgrService.run(lBreakPointStruct);
00194 stdair::EventType::EN_EventType lBreakPointType =
00195     lBreakPointStruct.getEventType();
00196
00198 BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
00199                          "The last event popped from the queue should be a "
00200                          "<< \"break point.\");
00201
00202 sevmgrService.run(lBreakPointStruct);
00203 lBreakPointType = lBreakPointStruct.getEventType();
00204
00206 BOOST_REQUIRE_MESSAGE (lBreakPointType == stdair::EventType::BRK_PT,
00207                          "The last event popped from the queue should be a "
00208                          "<< \"break point.\");
00209
00210 // Extract the corresponding date
00211 const stdair::DateTime_T& lBPDateTime =
00212     lBreakPointStruct.getEventTime ();
00213 const stdair::Date_T& lBPDate =
00214     lBPDateTime.date();
00215
00217 const stdair::Date_T lExpectedBPDate (2011, boost::gregorian::May, 14);
00218 BOOST_REQUIRE_MESSAGE (lBPDate == lExpectedBPDate,
00219                          "Date of the second break point popped from the queue:
00220
00221                          << lBPDate << ". Should be: "
00222                          << lExpectedBPDate << ".");
00223
00223 // DEBUG
00224 STDAIR_LOG_DEBUG ("End of the simulation");
00225
00226 // Close the log file
00227 logOutputFile.close();
00228 }
00229
00230 // End the test suite
00231 BOOST_AUTO_TEST_SUITE_END()
00232
00233

```