JMS

Test Suite Documentation

TABLE

1. JMS API versions

The test suite tests for JMS 1.1 compliance. The change of API (from JMS 1.0.2b to JMS 1.1) has introduced changes in the `Admin` interface, thus `Admin` implementations provided for JMS 1.0.2 are no longer compatible with the JMS 1.1 test suite.

But, of course, both version of the test suite will continue to coexist. A branch has been created in the CVS repository:

- the main branch of CVS is for JMS 1.1,
- the branch `jms-1_0_2b` is for JMS 1.0.2b.

Contributions and new tests are still welcome for both test suites.

2. Supported JMS providers

2.1. JMS 1.1 providers

At the present time, the supported JMS 1.1 providers are:

- JORAM 3.3 (http://www.objectweb.org/joram).
- SwiftMQ 4.0.0 (http://www.swiftmq.com/), provided by Andreas Mueller (mailto:am@iit.de).

2.2. JMS 1.0.2b providers

At the present time, the supported JMS 1.0.2b providers are:

- JORAM 3.0.x (http://www.objectweb.org/joram).
- AshnaMQ 2.0 (http://www.ashnasoft.com/) Standard Edition or Enterprise Edition, provided by Ashnasoft (mailto:support@ashnasoft.com).
- FioranoMQ 5.21 (http://fiorano.com/best_cgi/frame.cgi?target=products/fmq_tour.htm).
- Pramati Application Server 3.0 or Message Server (http://www.pramati.com/product/server30/index.htm), provided by Rajdeep Dua (mailto:rajdeep@pramati.com).

3. Installing the test suite

3.1. CVS repository

For getting the test suite from CVS:

```
    cvs
-d:pserver:anonymous@cvs.joram.debian-sf.objectweb.org:/cvsroot/joram
    login
```

When prompted for a password, simply press the **Enter** key.

Then you have to extract the **joram** module:

```
    cvs
-d:pserver:anonymous@cvs.joram.debian-sf.objectweb.org:/cvsroot/joram
    co tests
```

3.2. Package

The test suite can also be downloaded as a package at the download page (http://www.objectweb.org/joram/tests/download.html). The file is a Gzipped Tar file, so you will need both gzip and tar utilities to open it.

3.3. Description of the source tree

The root of the test suite source tree is the `tests` directory. It contains `build.xml` used by **Ant** to compile, generate the javadoc, start the tests, etc.

3.3.1. `src/` directory

This directory contains all source code of the suite. The tests classes are in the package `org.objectweb.jtests.jms.conform` and its subpackages. The `org.objectweb.jtests.jms.admin` package defines a simple interface for JMS administration as well as a factory to get a provider-specific implementation of this interface. The `org.objectweb.jtests.jms.framework` package holds classes extending **JUnit** to adapt it to JMS.

There is also a `org.objectweb.jtests.providers.admin` package which contains administration implementations for providers.

3.3.2. `doc/` directory

This directory contains the documentation (in PDF).

3.3.3. `lib/` directory

This directory contains all libraries needed to compile the tests.

3.3.4. `providers/` directory

This directory contains libraries needed to use JMS providers. For example, there is a joram subdirectory with all jar files used by JORAM.

3.3.5. `config/` directory

This directory contains configuration files used by the test suite. At the moment, there are two files:

`provider.properties`

`test.properties`

The first one states which provider is to be tested (more on that later). The second one states all properties which can be configured for the test suite. At the moment there is only a `timeout` property, which sets the timeout value used when receiving messages in synchronous mode.

3.3.6. Other directories

Other directories will be created when compiling, testing or generating the javadoc.

4. Compiling and using the test suite

4.1. Prerequisites

To compile or use the test suite, **Ant** (http://jakarta.apache.org/ant) is needed. The Ant 1.4.1 version is recommended. You will also need the optional targets library, `jakarta-ant-1.4.1-optional.jar` which defines `junit` target and is also available on Ant website.

Once Ant environment is properly installed (see Ant documentation), you have to put both `jakarta-ant-1.4.1-optional.jar` you've just downloaded and `junit.jar` (which is in the `lib` directory) in `$ANT_HOME/lib` directory.

4.2. Javadoc

To generate the Javadoc documentation of the suite, in the `tests` directory, type:

`ant javadoc`

All generated files will be put in the `javadoc` directory.

The javadoc documentation contains a description for each test methods of the JMS specifications features which is tested by it.

4.3. Compilation of the test suite

In the `tests` directory, simply type:

`ant compile`

to do so.

4.4. Compilation of the providers

There is only one class for each provider to compile: the implementation of a simple administration interface. Each JMS providers has its own way to create JMS administrated objects (connection factories and destinations). To remove this dependance from the test suite, it uses a simple `Admin` interface to create these objects. The implementation of the interface is discovered at runtime by a property put in the `config/provider.properties` file which indicates which class implementing the `Admin` interface to use.

To compile these provider specific classes, in the `tests` directory, type:

`ant compile.<provider>`

`<provider>` being of one the providers which implements the Admin interface.

4.4.1. JORAM

To compile JORAM `Admin` implementation, type in the `tests` directory:

`ant compile.joram`

4.4.2. AshnaMQ

To compile AshnaMQ, first you have to add the `AshnaClient.jar` file in the `providers/ashnasoft` directory (this file can be found in the `lib` directory of your installation of AshnaMQ). Then, type in the `tests` directory:

`ant compile.ashnasoft`

4.4.3. FioranoMQ

To compile FioranoMQ `Admin` implementation, you have first to install and properly setup it (for more information see FioranoMQ web site, http://fiorano.com/best_cgi/frame.cgi?target=products/fmq_tour.htm) then add fioranoMQ libraries in the `providers/fiorano` directory and type in the `tests` directory:

`ant compile.fiorano`

Provider specific classes are not compiled with the test suite on purpose to avoid dependency of the suite to these providers.

4.4.4. Pramati

First install Pramati Message Server and copy the following jars to the `providers/pramati` directory from `$PRAMATI_INSTALL_ROOT/jms/lib/ext` and `$PRAMATI_INSTALL_ROOT/jms/lib/pramati`:

`jaas.jar`, jce1_2_1.jar , `jcert.jar`, `jmxri.jar`, `jndi.jar`, `jnet.jar`, `jsse.jar`, `pramati_jms_client.jar`, `pramati_jms_xa.jar`, `pramati_jmxconn.jar`, `pramati_naming.jar`, `pramati_security.jar`, `pramati_serveradmin.jar`, `pramati_spi.jar`, `pramati_util.jar`, `xml.jar`.

and type in the `tests` directory:

`ant compile.pramati`

4.4.5. SwiftMQ

First install SwiftMQ and copy all the jar files from SwiftMQ's `jars` directory into `providers/swiftmq` and type in the `tests` directory:

`ant compile.swiftmq`

5. Starting the tests

5.1. Starting a JMS provider

Before starting the test, provider specific setup has to be made to start a given server. Since there are too many ways to do so, this step is not included in the suite process.

5.1.1. JORAM

Go to the `providers/joram` directory and type:

`ant server`

5.1.2. AshnaMQ

Let say that ASMQ_HOME is the home directory of AshnaMQ, go to `$ASMQ_HOME/bin` directory and type:

`server.sh` (on Unix)

`server.bat` (on Win32)

5.1.3. FioranoMQ

Let say that FIORANO_HOME is the home directory of FioranoMQ, go to `$FIORANO_HOME/scripts` directory and type :

`runkernel.sh`

5.1.4. Pramati

Please refer to Pramati Message Server's documentation: http://www.pramati.com/docstore/1230006/help/index1.htm

5.1.5. SwiftMQ

Let say that SWIFTMQ_HOME is the home directory of SwiftMQ, go to `$SWIFTMQ_HOME/scripts<platform>` directory and type :

`smqr1`

5.2. Three ways to test

There are three ways to use the test suite. Once the JMS provider is started, you can type one of these three commands :

`ant batchtest`

`ant junit.gui`

`ant test -Dtest=<test class name>`

The first one starts all the tests of the test suite. The second one starts JUnit graphic interface and let you chose which tests you want to run. The last one let you run one test case given the name of this case (for example `org.objectweb.jtests.jms.message.MessageBodyTest`) if you're only interested to test specific JMS features.

5.3. Test Suite Report

For all test case classes, an XML report is generated. Then thanks to the `junitreport` task, a Javadoc like report is generated in the `report/html` directory which sums up the tests with messages for failed ones.

This report generation is done automatically if you started tests either by ***`ant batchtest`*** or ***`ant test -Dtest=<test class name>`***.