Mobicents JAIN SLEE
SleeConnectivity Example User Guide

by Bartosz Baranowski and Eduardo Martins

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.

> Press **Ctrl+Alt+F1** to switch to the first virtual terminal. Press **Ctrl+Alt+F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type `ssh` *`username@domain.name`* at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.
>
> The `mount -o remount` *`file-system`* command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.
>
> To see the version of a currently installed package, use the `rpm -q` *`package`* command. It will return a result as follows: *`package-version-release`*.

Note the words in bold italics above username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as

a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books         Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads       images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
  public static void main(String args[])
     throws Exception
  {
    InitialContext iniCtx = new InitialContext();
    Object       ref   = iniCtx.lookup("EchoBean");
    EchoHome      home  = (EchoHome) ref;
    Echo        echo  = home.create();

    System.out.println("Created Echo");

    System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
  }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the *Issue Tracker* [http://code.google.com/p/mobicents/issues/list], against the product **Mobicents JAIN SLEE SleeConnectivity Example**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_SleeConnectivity_EXAMPLE_User_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Introduction to Mobicents JAIN SLEE SleeConnectivity Example

This example demonstrates the usage of SLEEConnection API, present in JAIN SLEE 1.1 specification, by a Java EE application, to fire events into a Mobicents JAIN SLEE container. The Java EE application and the Mobicents JAIN SLEE container may be in same or different JVM.

Setup

2.1. Pre-Install Requirements and Prerequisites

Ensure that the following requirements have been met before continuing with the install.

2.1.1. Hardware Requirements

The Example doesn't change the Mobicents JAIN SLEE Hardware Requirements, refer to Mobicents JAIN SLEE documentation for more information.

2.1.2. Software Prerequisites

The Example requires Mobicents JAIN SLEE properly set. In case of using separated JVMs to host the JAva EE and JAIN SLEE applications, the Mobicents JAIN SLEE Remote SLEE Connection Tool must be deployed in the Java EE server, and properly configured, to be able to interact with the remote JAIN SLEE container.

2.2. Mobicents JAIN SLEE SleeConnectivity Example Source Code

2.2.1. Release Source Code Building

1. **Downloading the source code**

   > **Important**
   >
   > Subversion is used to manage its source code. Instructions for using Subversion, including install, can be found at *http://svnbook.red-bean.com*

   Use SVN to checkout a specific release source, the base URL is http://mobicents.googlecode.com/svn/tags/servers/jain-slee/2.x.y/examples/slee-connectivity, then add the specific release version, lets consider 2.4.0.CR1.

   ```
   [usr]$ svn co http://mobicents.googlecode.com/svn/tags/servers/jain-slee/2.x.y/examples/slee-connectivity/2.4.0.CR1 slee-example-slee-connectivity-2.4.0.CR1
   ```

2. **Building the source code**

> **Important**
>
> Maven 2.0.9 (or higher) is used to build the release. Instructions for using
> Maven2, including install, can be found at *http://maven.apache.org*

Use Ant to build the binary.

```
[usr]$ cd slee-example-slee-connectivity-2.4.0.CR1
[usr]$ mvn install
```

Once the process finishes you should have the JAIN SLEE `deployable-unit` jar file in the `slee/du/target` directory, and JMX Client in the javaee/beans/target directory.

2.2.2. Development Trunk Source Building

Similar process as for *Section 2.2.1, "Release Source Code Building"*, the only change is the SVN source code URL, which is http://mobicents.googlecode.com/svn/trunk/servers/jain-slee/examples/slee-connectivity.

2.3. Installing Mobicents JAIN SLEE SleeConnectivity Example

To install just copy the JMX Client -bean directory to the deploy directory of the JBoss AS5, and the JAIN SLEE deployable unit to the deploy directory of the JAIN SLEE container.

2.4. Uninstalling Mobicents JAIN SLEE SleeConnectivity Example

To uninstall simply delete the JMX Client -bean directory and JAIN SLEE deployable unit jar copied in the install procedures.

2.5. Configuring the Java EE server to interact with a remote JAIN SLEE

In this setup the Mobicents JAIN SLEE Remote SLEE Connection Tool must be deployed and properly configured, in the Java EE server, please refer to its User Guide for instructions on how to install and configure it.

Design Overview

The SleeConnectivity Example is composed by two parts: a JMX Bean to be deployed in a JBoss AS5 instance, the Java EE application which will be used to fire JAIN SLEE event to the JAIN SLEE container; and a JAIN SLEE application, to be deployed in the JAIN SLEE container, which will handle the events fired, printing information about it in the container's console.

3.1. Design Overview: SLEE Service

JAIN SLEE service is simplest of all. It performs very basic actions when handling each received event:

- detach from activity where the event was fired, this will allow the container to end it implicitly, since it becomes unreferenced.

- log the message contained in the event, which can be seen in the container's console.
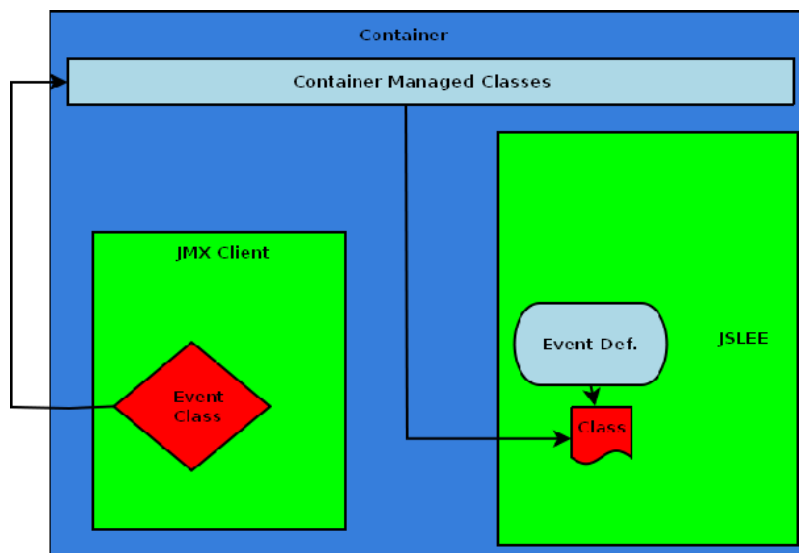
3.2. Design Overview: JMX Bean

JMX Client fires events into JAIN SLEE container, using the standard SLEEConnection API.

There are two possible scenarios of client and server deployment:

colocated
> JMX Bean and JAIN SLEE run in the same JVM. In this case both parts are deployed in same JBoss AS, and no additional tools are needed.
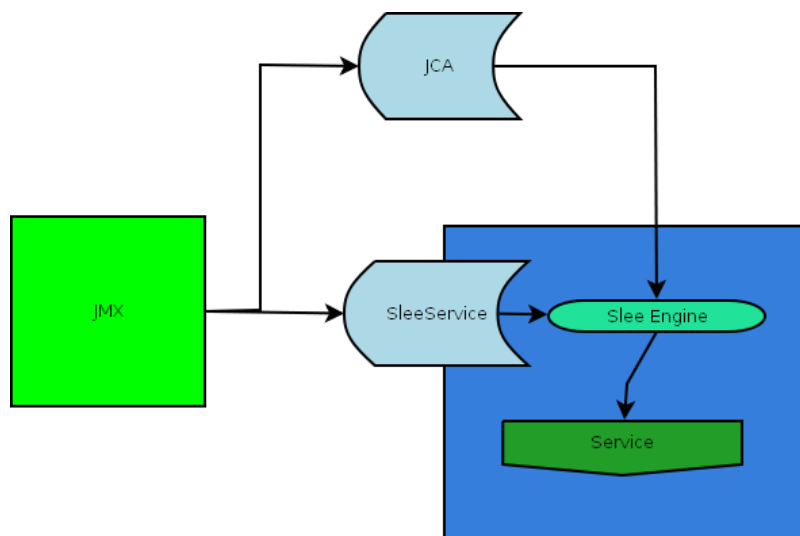

Container class sharing

separate
> In this case JMX Client and JSLEE container run in different JVMs. The Mobicents JAIN SLEE Remote SLEE Connection Tool, which uses JCA, must be deployed in the Java EE container.

Diagram below depicts how JMX Client fires event into JSLEE:

Container class sharing

Source Code Overview

> **Important**
>
> To obtain the example's complete source code please refer to *Section 2.2, "Mobicents JAIN SLEE SleeConnectivity Example Source Code"*.

4.1. Source Code Overview: Sbb

> **Note**
>
> For full descirption of XML please refer to simpler examples, like sip-wakeup

SBB descriptor is very simple. Its only purpose is to define SBB abstract class and event handler. Abstract class is defined as follows:

```xml
<sbb-name>SleeConnectivitySbb</sbb-name>
<sbb-vendor>org.mobicents</sbb-vendor>
<sbb-version>1.0</sbb-version>

<sbb-classes>
  <sbb-abstract-class>
    <sbb-abstract-class-name>
        org.mobicents.slee.service.SleeConnectivitySbb
    </sbb-abstract-class-name>
  </sbb-abstract-class>
</sbb-classes>
```

Handler definition looks as follows:

```xml
<event event-direction="Receive" initial-event="True">
  <event-name>CustomEvent</event-name>
  <event-type-ref>
   <event-type-name>org.mobicents.slee.service.connectivity.Event_1</event-type-name>
    <event-type-vendor>org.mobicents</event-type-vendor>
```

```
        <event-type-version>1.0</event-type-version>
      </event-type-ref>
      <initial-event-select variable="ActivityContext" />
    </event>
```

4.2. Source Code Overview: JMX Client

JMX Client is defined by two elements: XML descriptor and POJO class.

XML descriptor is very simple. It looks as follows(`jboss-beans.xml`):

```xml
<?xml version="1.0" encoding="UTF-8"?>

<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:jboss:bean-deployer:2.0">

  <bean name="SleeConnectionTest"
    class="org.mobicents.example.slee.connection.SleeConnectionTest">
    <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(
      name="org.mobicents.slee:name=SleeConnectivityExample",exposedInterface=
      org.mobicents.example.slee.connection.SleeConnectionTestMBean.class
        ,registerDirectly=true)
    </annotation>
  </bean>

</deployment>
```

Descriptor has following elements present:

bean
    definition of POJO bean with human readable name and class

annotation
    marks POJO as JMX exposed bean

property
    sets value of java bean property with matching name

demand

creates requirement - bean with name passed in this tag must be present for declared bean to be deployed

Client contract is defined by implemented interface class, that is `org.mobicents.example.slee.connection.SleeConnectionTestMBean`

public void fireEvent(String message);

Fires event using to JAIN SLEE container. The event will contain the specified message parameter.

Concrete implementation can be found in `org.mobicents.example.slee.connection.SleeConnectionTest` class.

```java
public void fireEvent(String messagePassed) {

// depending on deployment it does following:
// 1. lookup RA and make RMI calls through it
// 2. lookup local Bean, which makes direct calls to container!
logger.info("Attempting call to SleeConnectionFactory.");
try {

    InitialContext ic = new InitialContext();
    // this is call to local JNDI space, private, it cant be accessed
    // from other JVM
    SleeConnectionFactory factory = (SleeConnectionFactory) ic
            .lookup("java:/MobicentsConnectionFactory");

    SleeConnection conn1 = null;
    try {
        conn1 = factory.getConnection();

        ExternalActivityHandle handle = conn1.createActivityHandle();

        EventTypeID requestType = conn1.getEventTypeID(eventName,
                eventVendor, eventVersion);
        CustomEvent customEvent = new CustomEvent();
        customEvent.setMessage(messagePassed);
        logger.info("The event type is: " + requestType);

        conn1.fireEvent(customEvent, requestType, handle, null);
    } finally {
```

```java
            if (conn1 != null)
                conn1.close();
        }

    } catch (Exception e) {
        logger.error("Exception caught in event fire method!", e);

    }
}
```

Running the Example

To run example, start `JBOSS AS` container and access its `jmx-console`. Open `http://${jboss.bind.address}:port/jmx-console` in browser.

> **Note**
>
> `jboss.bind.address` and `port` depends on how container is started, by default its equal to: `http://127.0.0.1:8080/jmx-console`

Lookup bean with name equal to: `org.mobicents.slee:name=SleeConnectivityExample` and invoke operations. Pass distinct argument to operation. If operation is successful, passed value should be shown in Mobicents JAIN SLEE Console

Traces and Alarms

6.1. Tracers

TODO Enumerate the Tracers used by the Example.

6.2. Alarms

Example does not rise any alarms.

Appendix A. Revision History

Revision History

Revision 1.0	Tue Dec 30 2009	EduardoMartins

Creation of the Mobicents JAIN SLEE SleeConnectivity Example User Guide.

Index

F
feedback, viii