Mobicents Remote SLEE Connection Tool 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 *Issue Tracker* [http://code.google.com/p/mobicents/issues/list], against the product **Mobicents JAIN SLEE**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_JCA_Remote_Slee_Connection_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 Remote SLEE Connection Tool

Mobicents Remote SLEE Connection Tool is a Java EE Connector Architecture (JCA) tool, which can be used to interact with Mobicents JAIN SLEE remotely.

The JAIN SLEE specification includes an API for interaction with JAIN SLEE container, with the interface `javax.slee.connection.SleeConnectionFactory` upfront, which allows users to create connections to a specific JAIN SLEE instance, and use that to fire events.

Mobicents Remote SLEE Connection Tool exposes `javax.slee.connection.SleeConnectionFactory` in the local JNDI tree.

For further information about SLEE connection factory please refer to JAIN SLEE 1.1 specification, Appendix F.

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

N/A

2.1.2. Software Prerequisites

The Remote SLEE Connection Tool is a JCA connector, thus it requires a Java EE platform to run it.

2.2. Mobicents Remote SLEE Connection Tool 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/tools/remote-slee-connection, then add the specific release version, lets consider 2.4.0.FINAL.

```
[usr]$ svn co http://mobicents.googlecode.com/svn/tags/servers/jain-slee/2.x.y/tools/remote-slee-connection/2.4.0.FINAL remote-slee-connection-2.4.0.FINAL
```

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 Maven to build the deployable unit binary.

```
[usr]$ cd remote-slee-connection-2.4.0.FINAL
[usr]$ mvn install
```

Once the process finishes you should have the `JCA` mobicents-slee-remote-connection.rar directory artifact in the `target/mobicents-slee-remote-connection.rar-beans` directory. Deploying the JCA connector depends on the Java EE platform used, in case of JBoss Application Server simply copy the directory artifact to the platform's deploy 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/tools/remote-slee-connection.

2.3. Installing Mobicents Remote SLEE Connection Tool

To install the tool to the JBoss Application Server simply copy the provided `mobicents-slee-remote-connection.rar` directory into the `deploy` directory of the target server profile.

For other Java EE platforms please refer to its documentation on how to deploy the JCA connector.

2.4. Uninstalling Mobicents Remote SLEE Connection Tool

To uninstall the tool from the JBoss Application Server simply delete the `mobicents-slee-remote-connection.rar` directory from the `deploy` directory of the target server profile.

For other Java EE platforms please refer to its documentation on how to undeploy the JCA connector.

Configurating and Using Mobicents Remote SLEE Connection Tool

3.1. Configuring Mobicents Remote SLEE Connection Tool

The only configuration needed is setting up the address of the JAIN SLEE container to interact, this is done through the config-property named SleeJndiName inside `slee-ds.xml` file inside the rar archive or directory. ${jboss.bind.address} should be replaced with the hostname used by the JAIN SLEE container.

3.2. Using Mobicents Remote SLEE Connection Tool

The tool installs the SLEE Connection Factory in JNDI, below is example code which retrieves it and uses it to fire an event into the JAIN SLEE container.

```
// retrieves JNDI context
InitialContext context = new InitialContext();

// retrieves the connection factory from JNDI
        SleeConnectionFactory factory = (SleeConnectionFactory) context.lookup("java:/MobicentsConnectionFactory");

// creates a connection to the SLEE container
SleeConnection connection = factory.getConnection();

// creates the activity handle which will be used to fire the event
ExternalActivityHandle handle = connection.createActivityHandle();

// retrieves the event type ID
EventTypeID eventTypeID = connection.getEventTypeID("CustomEvent", "...", "1.0");

// creates the event object
CustomEvent eventObject = new CustomEvent();

// fires the event in the remote SLEE container
connection.fireEvent(eventObject, eventTypeID, handle, null);

// closes the connection
```

```
    connection.close();
```

Appendix A. Revision History

Revision History

Revision 1.0 Tue Dec 22 2009 EduardoMartins

Creation of the Mobicents Remote SLEE Connection Tool User Guide.

Index

F