JBoss Communications JAIN SLEE XMPP Resource Adaptor User Guide

by Eduardo Martins, Bartosz Baranowski, and Alexandre Mendonça

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://bugzilla.redhat.com/bugzilla/], against the product **JBoss Communications JAIN SLEE XMPP Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_XMPP_RA_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 JBoss Communications JAIN SLEE XMPP Resource Adaptor

The XMPP Resource Adaptor provides interaction with XMPP resources, supporting XMPP Client and Component connections. The RA adapts the well known and open source `SMACK 3.x` XMPP Client API, from JIVE Software, extending it with XMPP Component connections. For further information about the `SMACK` API refer to its website at *http://www.igniterealtime.org/projects/smack/*.

Resource Adaptor Type

The Resource Adaptor Type is the interface which defines the contract between the RA implementations, the SLEE container, and the Applications running in it.

The name of the RA Type is `XMPPResourceAdaptorType`, its vendor is `org.mobicents` and its version is `2.0`.

2.1. Activities

The single activity object for XMPP Resource Adaptor is the `org.mobicents.slee.resource.xmpp.XmppConnection` interface. The activity represents a connection to an XMPP Server, of client or component type, where a sequence of incoming XMPP messages - the events - occur.

The XmppConnection activity starts when an SBB requests a new connection to be created, through the RA SBB Interface, and it ends when an SBB uses the same interface to request the closing of the connection. The activity also gives access to the underlying connection on the SMACK API, through the method named `getConnection()`. With that object the SBB is able to completely interact with the resource.

For further information about the `SMACK` API refer to its website at *http://www.igniterealtime.org/projects/smack/*.

2.2. Events

The Events fired by XMPP Resource Adaptor represent an incoming message, received in a specific `XmppConnection` activity. The table below lists the Resource Adaptor Type event types.

Table 2.1. Events fired on the XmppConnection Activity

Name	Vendor	Version	Event Class	Description
org.jivesoftware.smack.packet.Message	org.jivesoftware.smack	1.0	org.jivesoftware.smack.packet.Message	An incoming `MESSAGE` XMPP stanza.
org.jivesoftware.smack.packet.Presence	org.jivesoftware.smack	1.0	org. jivesoftware.smack.packet.Presence	An incoming `PRESENCE` XMPP stanza.
org.jivesoftware.smackx.packet.DiscoverInfo	org.jivesoftware.smack	1.0	org.jivesoftware.smackx.packet.DiscoverInfo	An incoming `IQ` XMPP stanza related to a discovery of information about an XMPP Entity.

Name	Vendor	Version	Event Class	Description
org.jivesoftware.smackx.packet.DiscoverItems	org.jivesoftware.smack	1.0	org.jivesoftware.smackx.packet.DiscoverItems	An incoming `IQ` XMPP stanza related to a discovery of the items associated with an XMPP Entity.
org.jivesoftware.smack.packet.IQ	org.jivesoftware.smack	1.0	org.jivesoftware.smack.packet.IQ	An incoming and generic `IQ` XMPP stanza.
org.jivesoftware.smackx.packet.IQBasedAvatar	org.jivesoftware.smack	1.0	org.jivesoftware.smackx.packet.IQBasedAvatar	An incoming `IQ` XMPP stanza related to the IQ-Based Avatars XMPP extension.
org.jivesoftware.smack.packet.Registration	org.jivesoftware.smack	1.0	org.jivesoftware.smack.packet.Registration	An incoming `IQ` XMPP stanza, represents a registration packet. An empty GET query will cause the server to return information about it's registration support. SET queries can be used to create accounts or update existing account information.

Important

Spaces where introduced in `Name`, `Vendor` and `Event Class` column values, to correctly render the table. Please remove them when using copy/paste.

Important

More details for each event type may be obtained on the javadocs of each event type class.

2.3. Activity Context Interface Factory

The Resource Adaptor's Activity Context Interface Factory is of type `org.mobicents.slee.resource.xmpp.XmppActivityContextInterfaceFactory`. It allows the SBB to retrieve the `ActivityContextInterface` related with a specific `XmppConnection` activity object. The interface is defined as follows:

```
package org.mobicents.slee.resource.xmpp;

import javax.slee.ActivityContextInterface;
import javax.slee.FactoryException;
import javax.slee.UnrecognizedActivityException;

public interface XmppActivityContextInterfaceFactory {

    public ActivityContextInterface getActivityContextInterface(
            XmppConnection connection) throws NullPointerException,
            UnrecognizedActivityException, FactoryException;

}
```

2.4. Resource Adaptor Interface

The XMPP Resource Adaptor interface, of type `org.mobicents.slee.resource.xmpp.XmppResourceAdaptorSbbInterface` , is used by an SBB to interact with XMPP resources. It is defined as follows:

```
package org.mobicents.slee.resource.xmpp;

import java.util.Collection;

import org.jivesoftware.smack.XMPPException;
import org.jivesoftware.smack.packet.Packet;
```

```
public interface XmppResourceAdaptorSbbInterface {

 public XmppConnection getXmppConnection(String connectionId);

 public void sendPacket(String connectionID, Packet packet);

 public XmppConnection connectClient(String connectionID, String serverHost,
   int serverPort, String serviceName, String username,
   String password, String resource, Collection packetFilters)
   throws XMPPException;

 public XmppConnection connectComponent(String connectionID,
   String serverHost, int serverPort, String serviceName,
   String componentName, String componentSecret,
   Collection packetFilters) throws XMPPException;

 public void disconnect(String connectionID);
}
```

The `getXmppConnection(String)` method:

Retrieves the XMPP connection with the specified id.

The `sendPacket(String,Packet)` method:

Sends the specified XMPP packet using the specified connection Id.

The `connectClient(String, String, int, String, String, String, String, Collection)` method:

Creates and retrieves a new XMPP client connection to the XMPP server using the given service name on the given host and port and authenticates the client using the given username, password and resource. This connection will be identified with the specified connection Id. The `XmppConnection` returned is a new Activity in SLEE.

The `connectComponent(String, String, int, String, String, String, Collection)` method:

Creates and retrieves a new XMPP component connection to the XMPP server using the given service name on the given host and port and authenticates the component using the given component name and secret. The `XmppConnection` returned is a new Activity in SLEE.

The `disconnect(String)` method:

Disconnects the XMPP connection with the specified Id, terminating the activity and its related `ActivityContextInterface`.

2.5. Restrictions

The XMPP does not imposes any kind of restriction on the usage of the `SMACK` API java objects.

2.6. Sbb Code Examples

The following code examples shows how to use the Resource Adaptor Type for common functionalities

2.6.1. Connection Creation

The following code examples the creation of a client XMPP connection and attachment to the related activity context. The `xmppSbbInterface` object is the RA SBB Interface, while the object `xmppActivityContextInterfaceFactory` is the RA Activity Context Interface Factory, both obtained through the SBB JNDI environment.

```
try {
    XmppConnection connection = xmppSbbInterface.connectClient(
            connectionID, serviceHost, servicePort, serviceName,
            username, password, resource, Arrays
                    .asList(packetsToListen));
    xmppActivityContextInterfaceFactory.getActivityContextInterface(
            connection).attach(sbbContext.getSbbLocalObject());
} catch (XMPPException e) {
    tracer.severe("Connection to server failed!",e);
}
```

2.6.2. Message Sending

The following code examples the sending of a XMPP message, through the RA SBB Interface:

```
String connectionID = getXmppConnectionID();
String targetXmppUser = getTargetXmppUser();
Message msg = new Message(targetXmppUser, Message.Type.CHAT);
msg.setBody("Hello");
xmppSbbInterface.sendPacket(connectionID, msg);
```

Resource Adaptor Implementation

This chapter documents the XMPP Resource Adaptor Implementation details, such as the configuration properties, the default Resource Adaptor entities, and the JAIN SLEE 1.1 Tracers and Alarms used.

3.1. Configuration

The Resource Adaptor implementation does not currently have any configuration options.

3.2. Default Resource Adaptor Entities

There is a single Resource Adaptor Entity created when deploying the Resource Adaptor, named `XMPPRA`.

The `XMPPRA` entity is also bound to the Resource Adaptor Link Name `XMPPRA`. To use it in an SBB, add the following XML to its descriptor:

```
<resource-adaptor-type-binding>
 <resource-adaptor-type-ref>
  <resource-adaptor-type-name>
   XMPPResourceAdaptorType
  </resource-adaptor-type-name>
  <resource-adaptor-type-vendor>
   org.mobicents
  </resource-adaptor-type-vendor>
  <resource-adaptor-type-version>
   2.0
  </resource-adaptor-type-version>
 </resource-adaptor-type-ref>
 <activity-context-interface-factory-name>
  slee/resources/xmpp/2.0/acifactory
 </activity-context-interface-factory-name>
 <resource-adaptor-entity-binding>
  <resource-adaptor-object-name>
   slee/resources/xmpp/2.0/sbbinterface
  </resource-adaptor-object-name>
  <resource-adaptor-entity-link>
   XMPPRA
  </resource-adaptor-entity-link>
 </resource-adaptor-entity-binding>
</resource-adaptor-type-binding>
```

3.3. Traces and Alarms

3.3.1. Tracers

Each Resource Adaptor Entity uses a single JAIN SLEE 1.1 Tracer, named `XmppResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=XMPPRA]`.

3.3.2. Alarms

No alarms are set by this Resource Adaptor.

Setup

4.1. Pre-Install Requirements and Prerequisites

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

4.1.1. Hardware Requirements

The RA hardware requirements don't differ from the underlying JBoss Communications JAIN SLEE requirements, refer to its documentation for further information.

4.1.2. Software Prerequisites

The RA requires JBoss Communications JAIN SLEE properly set.

4.2. JBoss Communications JAIN SLEE XMPP Resource Adaptor Source Code

4.2.1. 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 ?, then add the specific release version, lets consider 2.4.1-SNAPSHOT.

```
[usr]$ svn co ?/2.4.1-SNAPSHOT slee-ra-xmpp-2.4.1-SNAPSHOT
```

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 slee-ra-xmpp-2.4.1-SNAPSHOT
[usr]$ mvn install
```

Once the process finishes you should have the `deployable-unit` jar file in the `target` directory, if JBoss Communications JAIN SLEE is installed and environment variable JBOSS_HOME is pointing to its underlying JBoss Enterprise Application Platform directory, then the deployable unit jar will also be deployed in the container.

4.2.2. Development Trunk Source Building

Similar process as for *Section 4.2.1, "Release Source Code Building"*, the only change is the SVN source code URL, which is NOT AVAILABLE.

4.3. Installing JBoss Communications JAIN SLEE XMPP Resource Adaptor

To install the Resource Adaptor simply execute provided ant script `build.xml` default target:

```
[usr]$ ant
```

The script will copy the RA deployable unit jar to the `default` JBoss Communications JAIN SLEE server profile deploy directory, to deploy to another server profile use the argument `-Dnode=`.

4.4. Uninstalling JBoss Communications JAIN SLEE XMPP Resource Adaptor

To uninstall the Resource Adaptor simply execute provided ant script `build.xml undeploy` target:

```
[usr]$ ant undeploy
```

The script will delete the RA deployable unit jar from the `default` JBoss Communications JAIN SLEE server profile deploy directory, to undeploy from another server profile use the argument `-Dnode=`.

Clustering

The XMPP Resource is not currently cluster aware. It does not provide failover for the resources it manages, i.e., XMPP connections.

Appendix A. Revision History

Revision History
Revision 1.0 Tue Dec 30 2009 EduardoMartins
Creation of the JBoss Communications JAIN SLEE XMPP RA User Guide.

Index

F