Mobicents JAIN SLEE Java Call Control (JCC) Resource Adaptor User Guide

by Bartosz Baranowski, Eduardo Martins, and Oleg Kulikov

Preface v
1. Document Conventions v
1.1. Typographic Conventions v
1.2. Pull-quote Conventions vii
1.3. Notes and Warnings vii
2. Provide feedback to the authors! viii
1. Introduction to Mobicents JAIN SLEE JCC Resource Adaptor 1
2. Resource Adaptor Type 3
3. Resource Adaptor Implementation 5
3.1. Configuration 5
3.2. Default Resource Adaptor Entities 6
3.3. Traces and Alarms 7
3.3.1. Tracers 7
3.3.2. Alarms 7
4. Setup 9
4.1. Pre-Install Requirements and Prerequisites 9
4.1.1. Hardware Requirements 9
4.1.2. Software Prerequisites 9
4.2. Mobicents JAIN SLEE JCC Resource Adaptor Source Code 9
4.2.1. Release Source Code Building 9
4.2.2. Development Trunk Source Building 10
4.3. Installing Mobicents JAIN SLEE JCC Resource Adaptor 10
4.4. Uninstalling Mobicents JAIN SLEE JCC Resource Adaptor 10
A. Revision History 13
Index 15

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 JCC Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_JCC_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 Mobicents JAIN SLEE JCC Resource Adaptor

The JCC API defines a programming interface to next-generation converged networks in terms of an abstract, object-oriented specification. As such it is designed to hide the details of the specifics of the underlying network architecture and protocols from the application programmer to the extent possible. Thus the network may consist of the PSTN, a packet (IP or ATM) network, a wireless network, or a combination of these, without affecting the development of services using the API. The API is also independent of network signaling and transport protocols. Thus the network may be using various call control protocols and technologies, for example, SGCP, MGCP, SIP, H.323, ISUP, DSS1/Q.931, and DSS2/Q.2931, without the explicit knowledge of the application programmer. Indeed, different legs of a call may be using different signaling protocols and be on different underlying networks.

The JAIN JCC Specification defines an API which allows for the rapid creation and deployment of dynamic telephony services into a Java telephony platform. Traditionally, telephony applications require costly resources to develop, test, and deploy. A JAIN software component written to the JCC API can be rapidly developed, tested, and integrated on a variety of platforms with access to numerous tools and utilities. A JAIN cross-platform solution gives the Carriers, Service Providers, and Network Equipment Providers a consistent, open environment where they can develop and deploy telephony services.

The JAIN JCC Specification provides an interface to underlying call processing platforms supplied by platform implementers. It is expected that JAIN JCC platform providers will support a variety of lower-layer signaling, coordination and transaction protocols, such as MGCP, SIP, H.323, ISUP, TCAP, etc., in order to implement the facilities provided via the JCC API. However, the JCC API shield application developers from the specifics of the various networks and protocols.

Resource Adaptor Type

The JAIN JCC Resource Adaptor Type is specified in Appendix C of the JAIN SLEE 1.1 Specification. The specification can be freely downloaded from *http://jcp.org/aboutJava/communityprocess/final/jsr240/index.html*

Resource Adaptor Implementation

The RA implementation is adoptation of Java Call Control API (JCC 1.1) for the requirements of Jain SLEE. Any specific JCC providers can be injected into the Resource Adaptor. The Mobicents development teams provides implementation of the JCC for Customised Applications for Mobile networks Enhanced Logic (CAMEL) based on Mobicents SS7 solution.

3.1. Configuration

The Resource Adaptor supports configuration only at Resource Adaptor Entity creation time, the following table enumerates the configuration properties:

Table 3.1. Resource Adaptor's Configuration Properties

Property Name	Description	Property Type	Default Value
provider	Specifies JccPeer object given a fully qualified classname of the class which implements the JccPeer object or provider acronym.	java.lang.String	
config	The name of the property file which contains vendor/ protocol specific configuration	java.lang.String	

Configuration option(`provider`) supports following values(aside class full name):

TCP
: Following configuration options are available for this driver:

Table 3.2. config.properties values for TCP SCCP provider

Property Name	Description	Property Type	Default Value
driver	Configures stream driver of SCCP layer. Currently only TCP is supported.	java.lang.String	TCP
server.ip	Driver configuration options. Specifies TCP address to which driver will connect.	java.lang.String	127.0.0.1

Property Name	Description	Property Type	Default Value
server.port	Specifies server port to which driver will connect.	java.lang.String	1345
sccp.opc	Default OPC used by SCCP layer for originating messages.	java.lang.String	0
sccp.dpc	Default DPC used by SCCP layer for originating messages.	java.lang.String	0
sccp.sls	Default SLS used by SCCP layer for originating messages.	java.lang.String	0
sccp.ssi	Default SSI used by SCCP layer for originating messages.	java.lang.String	0

Important

JAIN SLEE 1.1 Specification requires values set for properties without a default value, which means the configuration for those properties are mandatory, otherwise the Resource Adaptor Entity creation will fail!

3.2. Default Resource Adaptor Entities

There is a single Resource Adaptor Entity created when deploying the Resource Adaptor, named JCCRA. The JCCRA entity uses the default Resource Adaptor configuration, specified in *Section 3.1, "Configuration"*.

The JCCRA entity is also bound to Resource Adaptor Link Name JCCRA, 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>JCC-1.1-RA</resource-adaptor-type-name>
      <resource-adaptor-type-vendor>javax.csapi.cc.jcc</resource-adaptor-type-vendor>
      <resource-adaptor-type-version>1.1</resource-adaptor-type-version>
   </resource-adaptor-type-ref>
   <activity-context-interface-factory-name>
      slee/resources/jcc/1.1/acifactory
   </activity-context-interface-factory-name>
```

```
        <resource-adaptor-entity-binding>
          <resource-adaptor-object-name>
            slee/resources/jcc/1.1/provider
          </resource-adaptor-object-name>
          <resource-adaptor-entity-link>
            JCCRA
          </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 `JCCResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=JCCRA]`

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 Resource Adaptor is adaptation of Java Call Control API to the requirements of the Jain SLEE. It does not create any specific hardware requirements however the underlying platform implementation may have specific hardware requirements. See JCC provider's documentation for details.

4.1.2. Software Prerequisites

The RA requires Mobicents JAIN SLEE properly set.

4.2. Mobicents JAIN SLEE JCC 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 http://mobicents.googlecode.com/svn/tags/servers/jain-slee/2.x.y/resources/jcc, then add the specific release version, lets consider 2.1.2.FINAL.

```
[usr]$ svn co http://mobicents.googlecode.com/svn/tags/servers/jain-slee/2.x.y/resources/jcc/2.1.2.FINAL slee-ra-jcc-2.1.2.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 slee-ra-jcc-2.1.2.FINAL
[usr]$ mvn install
```

Once the process finishes you should have the `deployable-unit` jar file in the `target` directory, if Mobicents JAIN SLEE is installed and environment variable JBOSS_HOME is pointing to its underlying JBoss Application Server 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 http://mobicents.googlecode.com/svn/trunk/servers/jain-slee/resources/jcc.

4.3. Installing Mobicents JAIN SLEE JCC 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` Mobicents JAIN SLEE server profile deploy directory, to deploy to another server profile use the argument `-Dnode=`.

4.4. Uninstalling Mobicents JAIN SLEE JCC 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` Mobicents JAIN SLEE server profile deploy directory, to undeploy from another server profile use the argument `-Dnode=`.

Appendix A. Revision History

Revision History

Revision 1.0	Sun Jun 13 2010	OlegKulikov

Creation of the Mobicents JAIN SLEE SIP11 RA User Guide.

Index

F

feedback, viii