Preface

Table of Contents

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 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.

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 kbd:[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 kbd:[Enter] to execute the command.
>
> Press to switch to the first virtual terminal. Press 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 menu: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 btn:[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 menu:Applications > Accessories > Character Map[] from the main menu bar. Next, choose menu:Search > Find[] from the `Character Map` menu bar, type the name of the character in the Search field and click btn:[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 btn:[Copy] button. Now switch back to your document and choose menu: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 menu:>[] shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select from the menu:Preferences[] sub-menu in the menu: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.

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"));
   }

}
```

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.

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 &THIS.ISSUE_TRACKER_URL;link:[Issue Tracker], against the product [app]` JAIN SLEE Resource Adaptor`, or contact the authors.

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

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.

1. Introduction to Restcomm JAIN SLEE SIP Resource Adaptor

This resource adaptor provides a SIP API for JAIN SLEE applications, adapting the JAIN SIP 1.2 specification. JAIN SIP 1.2 is a Java specification for the Session Initiation Protocol, as defined by the protocol specs done by the IETF. Both transaction and dialog layers of the SIP protocol are available, to support all types of SIP applications through a single API. Lower level applications, such as SIP Proxies or Registrars typically use the transaction layer exclusively, while UAC, UAS and B2BUA higher level SIP applications rely on the dialog layer.

Events represent SIP messages received by the SIP stack, or failure use cases such as timeouts. Unlike the base JAIN SIP 1.2 API, SIP Requests with different SIP methods are fired as different event types, the same happens for SIP Responses with status code. The events are fired on transaction or dialog activities.

The Activities are the SIP Transactions and Dialogs, which applications in the SLEE may use to send SIP Requests and Responses, and to receive the events related with incoming messages.

2. Resource Adaptor Type

The JAIN SIP 1.2 Resource Adaptor Type is specified in Appendix D of the JAIN SLEE 1.1 Specification. The specification can be freely downloaded from http://jcp.org/aboutJava/communityprocess/final/jsr240/index.html and includes Sbb code examples.

3. Resource Adaptor Implementation

The RA implementation uses the Restcomm JAIN SIP HA stack, an extension of the JAIN SIP Reference Implementation which provides high availability and fault tolerance. The stack is the result of the work done by Restcomm JAIN SLEE and SIP Servlets development teams, and source code is provided in all releases.

3.1. Configuration

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

Table 1. Resource Adaptor's Configuration Properties

Property Name	Description	Property Type	Default Value
javax.sip. IP_ADDRESS	the IP address to which the SIP stack should attach - if value is not specified the RA will use the underlying Java EE container's bind address address	java.lang.String	
javax.sip. OUTBOUND_PROXY	sets the outbound proxy of the SIP Stack. The format for this string is "ipaddress:port/transport" i.e. 129.1.22.333:5060/UDP. This property is optional	java.lang.String	

Property Name	Description	Property Type	Default Value
javax.sip. PORT	the port to which the SIP stack should listen	java.lang.Integer	5060
javax.sip. TRANSPORT	the list of supported transports, separated with ","	java.lang.String	UDP
org.mobicents. ha.javax.sip. BALANCERS	the list of SIP balancers, in the form of "HOST:PORT", separated by ";", it is only used if the heart beat service property is defined	java.lang.String	
org.mobicents. ha.javax.sip. LoadBalancer HeartBeating ServiceClassName	the name of the class responsible for the heart beats exchanged with the platform's SIP Balancer - if not specified the JAIN SIP HA stack won't use such feature	java.lang.String	
org.mobicents. javax.sip. LOOSE_DIALOG _VALIDATION	controls validation of CSeq number for dialog messages. Set to true allows out of sequence messages to be accepted.	java.lang.Boolean	

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

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 `SipRA`. The `SipRA` entity uses the default Resource Adaptor configuration, specified in Configuration.

The `SipRA` entity is also bound to Resource Adaptor Link Name `SipRA`. 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>
   JAIN SIP
  </resource-adaptor-type-name>
  <resource-adaptor-type-vendor>
   javax.sip
  </resource-adaptor-type-vendor>
  <resource-adaptor-type-version>
   1.2
  </resource-adaptor-type-version>
 </resource-adaptor-type-ref>

 <activity-context-interface-factory-name>
  slee/resources/jainsip/1.2/acifactory
 </activity-context-interface-factory-name>

 <resource-adaptor-entity-binding>
  <resource-adaptor-object-name>
   slee/resources/jainsip/1.2/provider
  </resource-adaptor-object-name>
  <resource-adaptor-entity-link>
   SipRA
  </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 `SipResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=SipRA]`.

3.3.2. Alarms

No alarms are set by this Resource Adaptor.

4. 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 hardware's main concern is RAM memory and Java Heap size, the more the better. For instance, while the underlying Restcomm JAIN SLEE may run with 1GB of RAM, 8GB is needed to achieve performance higher than 400 new calls per second.

Of course, memory is only needed to store the Resource Adaptor state, the faster the CPU more calls per second are supported, yet no particular CPU is a real requirement to use the RA.

4.1.2. Software Prerequisites

The RA requires Restcomm JAIN SLEE properly set.

4.2. Restcomm JAIN SLEE SIP Resource Adaptor Source Code

4.2.1. Release Source Code Building

1. Downloading the source code

Git is used to manage Restcomm JAIN SLEE source code. Instructions for downloading, installing and using Git can be found at http://git-scm.com/

Use Git to checkout a specific release source, the Git repository URL is https://github.com/Restcomm/jain-slee.sip, then switch to the specific release version, lets consider {this-version}.

```
[usr]$ git clone git@github.com:RestComm/jain-slee.sip.git
[usr]$ cd jain-slee-sip
[usr]$ git checkout tags/{this-version}
```

2. Building the source code

Maven 3.3.0 (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 resources/
[usr]$ mvn install
```

Once the process finishes you should have the `deployable-unit` jar file in the *target* directory, if Restcomm 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 Master Source Building

Similar process as for Release Source Code Building, the only change is the Git reference should be the `master`. The `git checkout tags/` command should not be performed. If already performed, the following should be used in order to switch back to the master:

```
[usr]$ git checkout master
```

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

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

5. Clustering

5.1. Failover

The SIP Stack used by the RA supports `ESTABLISHED` and `EARLY` dialog failover. This means that an application must be in charge of properly adapting its state machine to recover dialog state in case original transaction had not send or receive answer with code greater than `100`.

5.2. Load Balancing

The RA can be used with Restcomm SIP Load Balancer. The recommended version is {loadbalancer-version}.

5.2.1. Configuring the Resource Adaptor to be used with Restcomm SIP Load Balancer

There are three properties which define how the RA connects to Restcomm SIP Load Balancer:

org.mobicents.ha.javax.sip.BALANCERS

the property must be configured with the list of load balancer IP address and internal ports. As an example, suppose a single Restcomm SIP Load Balancer is running with IP `192.168.0.1` and internal port `5065`. The property would be set with a value of`192.168.0.1:5065`. To specify multiple balancers use `;` as a separator.

org.mobicents.ha.javax.sip.LoadBalancerHeartBeatingServiceClassName

this property is optional, and defines the class name of the HeartBeating service implementation. Currently, the only one available is `org.mobicents.ha.javax.sip.LoadBalancerHeartBeatingServiceImpl`

org.mobicents.ha.javax.sip.LoadBalancerElector

this property is optional, and defines the class of the load balancer elector from JAIN SIP HA Stack. The elector is used to define which load balancer will receive outgoing requests, which are out of dialog or in dialog with null state. Currently only one elector implementation is available, `org.mobicents.ha.javax.sip.RoundRobinLoadBalancerElector`, which, as the class name says, uses a round robin algorithm to select the balancer.

Appendix A: Revision History