

JBoss Communications JAIN SLEE Diameter Sh-Server Resource Adaptor User Guide

by Alexandre Mendonça

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 JBoss Communications JAIN SLEE Diameter Sh-Server Resource	
Adaptor	1
2. Resource Adaptor Type	3
2.1. Activities	3
2.2. Events	5
2.3. Activity Context Interface Factory	6
2.4. Resource Adaptor Interface	7
2.5. Restrictions	8
2.6. Sbb Code Examples	8
3. Resource Adaptor Implementation	11
3.1. Configuration	11
3.2. Default Resource Adaptor Entities	11
3.3. Traces and Alarms	12
3.3.1. Tracers	12
3.3.2. Alarms	12
4. Setup	13
4.1. Pre-Install Requirements and Prerequisites	13
4.1.1. Hardware Requirements	13
4.1.2. Software Prerequisites	13
4.2. JBoss Communications JAIN SLEE Diameter Sh-Server Resource Adaptor	
Source Code	13
4.2.1. Release Source Code Building	13
4.2.2. Development Trunk Source Building	14
4.3. Installing JBoss Communications JAIN SLEE Diameter Sh-Server Resource	
Adaptor	14
4.4. Uninstalling JBoss Communications JAIN SLEE Diameter Sh-Server Resource	
Adaptor	14
5. Clustering	17
5.1. Failover	17
5.2. Load Balancing	17
A. Revision History	19
Index	21

Preface

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/) [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 Of 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/) [http://bugzilla.redhat.com/bugzilla/], against the product **JBoss Communications JAIN SLEE Diameter Sh-Server Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_DIAMETER_SH_SERVER_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 Diameter Sh-Server Resource Adaptor

This resource adaptor provides a Diameter API for JAIN SLEE applications, according to Sh interface based on Diameter protocol.

Sh is a Diameter application that allows a Diameter server and a Diameter client to:

- download and update transparent and non-transparent user data;
- request and send notifications on changes on user data

Events represent Diameter Sh messages received by the Diameter stack. Different events types are specified for each Diameter request or answer. Since this RA only operates on the server-side, events are fired on server activities.

The Activities are defined by RA Type to ease use of RA. Activities represent Diameter session between two peers. SLEE applications use activities to create, send and receive messages.

Resource Adaptor Type

Diameter Sh-Server Resource Adaptor Type is defined by Mobicents team as part of effort to standardize RA Types.

2.1. Activities

Diameter Sh-Server Type 2.6.1.FINAL defines the following Activities:

`net.java.slee.resource.diameter.sh.server.ShServerActivity`

This type of activity represents server side of Sh session. User-Data-Request (UDR), Profile-Update-Request (PUR) and Subscribe-Notifications-Request (SNR) are received in this Activity and respective Answers are sent from it.

This activity type can only be created implicitly by the Resource Adaptor when receiving the above mentioned Requests. It ends once underlying Sh server session ends.

This activity sessions are simple Request-Answer sessions, meaning that once the Answer is sent or received it is terminated.

`net.java.slee.resource.diameter.sh.server.ShServerNotificationActivity`

This type of activity represents a subscription to User Data notifications for Sh session, based on User Identity. Push-Notification-Request (PNR) can be created and sent in this Activity, receiving the respective Answer (or timeout) later on this Activity.

This activity type can be created with call to the `createShServerNotificationActivity` method of `net.java.slee.resource.diameter.sh.server.ShServerProvider`. It ends once SNR with UNSUBSCRIBE as Subs-Req-Type for the affected User Identity is received.

All activities define methods required to properly function and expose necessary information to JAIN SLEE services. Sh Server Activity is defined as follows:

```
public UserDataAnswer createUserDataAnswer(byte[] userData);

public UserDataAnswer createUserDataAnswer(long resultCode,
    boolean isExperimentalResult);

public UserDataAnswer createUserDataAnswer();

public ProfileUpdateAnswer createProfileUpdateAnswer(long resultCode,
    boolean isExperimentalResult);
```

```
public ProfileUpdateAnswer createProfileUpdateAnswer();

public SubscribeNotificationsAnswer createSubscribeNotificationsAnswer();

public SubscribeNotificationsAnswer createSubscribeNotificationsAnswer(long resultCode,
    boolean isExperimentalResult);

public void sendUserDataAnswer(UserDataAnswer message) throws IOException;

public void sendProfileUpdateAnswer(ProfileUpdateAnswer message) throws IOException;

public void sendSubscribeNotificationsAnswer(SubscribeNotificationsAnswer message)
    throws IOException;
```

```
public UserDataAnswer createUserDataAnswer(byte[] userData);
```

This method creates a `UserDataAnswer` using the given parameter to populate the User-Data AVP. The Result-Code AVP is automatically set to SUCCESS. If there is no request of this type received, this method returns null.

```
public UserDataAnswer createUserDataAnswer(long resultCode, boolean isExperimentalResult);
```

This method creates a `UserDataAnswer` containing a Result-Code or Experimental-Result AVP populated with the given value. If *isExperimentalResultCode* is *true*, the *resultCode* parameter will be set as `ExperimentalResultAvp` AVP, if it is *false* the result code will be set in a Result-Code AVP. If there is no request of this type received, this method returns null.

```
public UserDataAnswer createUserDataAnswer();
```

This method creates an empty `UserDataAnswer` that will need to have AVPs set on it before being sent. If there is no request of this type received, this method returns null.

```
public ProfileUpdateAnswer createProfileUpdateAnswer(long resultCode, boolean
isExperimentalResult);
```

This method creates a `ProfileUpdateAnswer` containing a Result-Code or Experimental-Result AVP populated with the given value. If *isExperimentalResultCode* is *true*, the *resultCode* parameter will be set in a `ExperimentalResultAvp` AVP, if it is *false* the result code will be set in a Result-Code AVP. If there is no request of this type received, this method returns null.

```
public ProfileUpdateAnswer createProfileUpdateAnswer();
```

This method creates an empty `ProfileUpdateAnswer` that will need to have AVPs set on it before being sent. If there is no request of this type received, this method returns null.

```
public SubscribeNotificationsAnswer createSubscribeNotificationsAnswer();
```

This method creates an empty `SubscribeNotificationsAnswer` that will need to have AVPs set on it before being sent. If there is no request of this type received, this method returns null.

```
public SubscribeNotificationsAnswer createSubscribeNotificationsAnswer(long resultCode,
boolean isExperimentalResult);
```

This method creates a `SubscribeNotificationsAnswer` containing a `Result-Code` or `Experimental-Result` AVP populated with the given value. If there is no request of this type received, this method returns null.

```
public void sendUserDataAnswer(UserDataAnswer message) throws IOException;
```

This method sends a `UserDataAnswer` to the peer that sent the `UserDataRequest`.

```
public void sendProfileUpdateAnswer(ProfileUpdateAnswer message) throws IOException;
```

This method sends a `ProfileUpdateAnswer` to the peer that sent the `ProfileUpdateRequest`.

```
public void sendSubscribeNotificationsAnswer(SubscribeNotificationsAnswer message) throws
IOException;
```

This method sends a `SubscribeNotificationsAnswer` to the peer that sent the `SubscribeNotificationsRequest`.

Sh Server Notification Activity is defined as follows:

```
public ShServerMessageFactory getServerMessageFactory();
```

```
public void sendPushNotificationRequest(PushNotificationRequest message)
throws IOException;
```

```
public ShServerMessageFactory getClientMessageFactory();
```

This method returns a 'customized' message factory to manually create Sh Server Messages for this Activity.

```
public void sendPushNotificationRequest(PushNotificationRequest message) throws
IOException;
```

This method sends a `Push-Notification-Request` asynchronously.



Note

It is safe to type cast all the mentioned Diameter Activities to its super interface `net.java.slee.resource.diameter.base.DiameterActivity` defined in Diameter Base Activities section.

2.2. Events

Diameter Sh-Server Resource Adaptor Type declares all the Diameter Sh messages related to server operations.

The following tables show which events are fired on each activity.

Table 2.1. Events received on Sh Server Activity

Name	Vendor	Version	Class
net.java.slee.resource.diameter.sh.events.UserDataRequest	java.net	0.8	net.java.slee.resource.diameter.sh.events.UserDataRequest
net.java.slee.resource.diameter.sh.events.ProfileUpdateRequest	java.net	0.8	net.java.slee.resource.diameter.sh.events.ProfileUpdateRequest
net.java.slee.resource.diameter.sh.events.SubscribeNotificationsRequest	java.net	0.8	net.java.slee.resource.diameter.sh.events.SubscribeNotificationsRequest

Table 2.2. Events received on Sh Server Notification Activity

Name	Vendor	Version	Class
net.java.slee.resource.diameter.sh.events.PushNotificationAnswer	java.net	0.8	net.java.slee.resource.diameter.sh.events.PushNotificationAnswer



Important

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

2.3. Activity Context Interface Factory

The JBoss Communications Diameter Sh-Server Activity Context Interface Factory is defined as follows:

```
package net.java.slee.resource.diameter.sh.server;

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

public interface ShServerActivityContextInterfaceFactory {

    ActivityContextInterface getActivityContextInterface(ShServerActivity activity)
        throws UnrecognizedActivityException;
}
```

```

    ActivityContextInterface getActivityContextInterface(ShServerSubscriptionActivity activity)
        throws UnrecognizedActivityException;
}

```

2.4. Resource Adaptor Interface

The JBoss Communications Diameter Sh-Server Resource Adaptor SBB Interface provides SBBs with access to the Diameter objects required for creating and sending messages. It is defined as follows:

```

package net.java.slee.resource.diameter.sh.server;

import java.io.IOException;

import net.java.slee.resource.diameter.base.events.avp.DiameterIdentity;
import net.java.slee.resource.diameter.sh.DiameterShAvpFactory;
import net.java.slee.resource.diameter.sh.events.PushNotificationAnswer;
import net.java.slee.resource.diameter.sh.events.PushNotificationRequest;

public interface ShServerProvider {

    public ShServerMessageFactory getServerMessageFactory();

    public DiameterShAvpFactory getServerAvpFactory();

    public ShServerNotificationActivity createShServerNotificationActivity()
        throws CreateActivityException;

    public PushNotificationAnswer pushNotificationRequest(PushNotificationRequest message)
        throws IOException;

    public DiameterIdentity[] getConnectedPeers();

    public int getPeerCount();
}

```

```
public ShServerMessageFactory getServerMessageFactory();
```

This method returns a `ShServerMessageFactory` implementation to be used to create `DiameterMessage` objects.

`public DiameterShAvpFactory getServerAvpFactory();`

This method returns a `DiameterShAvpFactory` implementation to be used to create `DiameterAvp` objects.

`public ShServerNotificationActivity createShServerNotificationActivity() throws CreateActivityException;`

This method creates a new Sh Server Activity to send and receive notification related Diameter Sh messages.

`public PushNotificationAnswer pushNotificationRequest(PushNotificationRequest message) throws IOException;`

This method sends a synchronous `PushNotificationRequest` which will block until an answer is received from the peer.

`public DiameterIdentity[] getConnectedPeers();`

This method returns the identities of peers this Diameter resource adaptor is connected to.

`public int getPeerCount();`

This method returns the number of peers this Diameter resource adaptor is connected to.

2.5. Restrictions

Current Resource Adaptor Type has no defined restrictions.

2.6. Sbb Code Examples

Simple Server-Side Example that receives an User-Data-Request and sends an User-Data-Answer.

```
/* Method for receiving UDR and sending UDA with SUCCESS (code 2001) Result-Code. */
public void onUserDataRequest(UserDataRequest event, ActivityContextInterface aci) {
    if (tracer.isInfoEnabled()) {
        tracer.info("Received UDR:\r\n" + event);
    }

    UserDataAnswer answer = ((ShServerActivity) aci.getActivity()).
        createUserDataAnswer(2001, false);

    try {
        if (tracer.isInfoEnabled()) {
            tracer.info("Created UDA:\r\n" + answer);
        }

        ((ShServerActivity) aci.getActivity()).sendUserDataAnswer(answer);
    }
}
```



```

    catch (IOException e) {
        tracer.severe("Failed to send UDA.", e);
    }
}

```

Simple Server-Side Example that sends an Push-Notification-Request and receives an Push-Notification-Answer.

```

/* Method for handling timer event where a PNR created and sent */
public void onTimerEvent(TimerEvent event, ActivityContextInterface aci) {
    ShServerNotificationActivity activity = null;

    for (ActivityContextInterface _aci : this.getSbbContext().getActivities()) {
        if (_aci.getActivity() instanceof ShServerNotificationActivity) {
            activity = (ShServerNotificationActivity) _aci.getActivity();
            break;
        }
    }

    if (activity == null) {
        tracer.severe("Activity is null, with list: " + Arrays.toString(this.getSbbContext().
            getActivities()));
        return;
    }

    PushNotificationRequest request = activity.createPushNotificationRequest();
    tracer.info("Created PNR:\r\n" + request);

    try {
        UserIdentityAvp ui = avpFactory.createUserIdentity();
        ui.setPublicIdentity("sip:user@diameter.mobicents.org");
        request.setUserIdentity(ui);
        request.setUserData("<xml>Some secrete user xml</xml>");
        if(tracer.isInfoEnabled()) {
            tracer.info("Sending PNR:\r\n" + request);
        }
        activity.sendPushNotificationRequest(request);
    }
    catch (Exception e) {
        tracer.severe("Failed to send PNR.", e);
    }
}

```

```
}  
  
...  
  
/* Method for handling PNA messages. Just print the Result-Code AVP. */  
public void onPushNotificationAnswer(PushNotificationAnswer pna,  
    ActivityContextInterface aci) {  
    if (tracer.isInfoEnabled()) {  
        tracer.info("Push-Notification-Answer received. Result-Code[" + pna.getResultCode() + "].");  
    }  
}
```

Resource Adaptor Implementation

This RA uses the JBoss Communications Diameter Stack, an improvement over [jDiameter Stack](http://jdiameter.dev.java.net) [http://jdiameter.dev.java.net]. The stack is the result of the work done by JBoss Communications Diameter and jDiameter 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 3.1. Resource Adaptor's Configuration Properties

Property Name	Description	Property Type	Default Value
authApplicationIds	List of supported Authorization Application Ids in form of {vendor}: {application-id}, separated by comma ','	java.lang.String	10415:16777217



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 `DiameterShServer`. The `DiameterShServer` entity uses the default Resource Adaptor configuration, specified in [Section 3.1, "Configuration"](#).

The `DiameterShServer` entity is also bound to Resource Adaptor Link Name `DiameterShServer`, 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>Diameter Sh-Server</resource-adaptor-type-name>
    <resource-adaptor-type-vendor>java.net</resource-adaptor-type-vendor>
```

```
<resource-adaptor-type-version>0.8.1</resource-adaptor-type-version>
</resource-adaptor-type-ref>

<activity-context-interface-factory-name>
  slee/resources/JDiameterShServerResourceAdaptor/java.net/0.8.1/acif
</activity-context-interface-factory-name>

<resource-adaptor-entity-binding>
  <resource-adaptor-object-name>
    slee/resources/diameter-sh-server-ra-interface
  </resource-adaptor-object-name>
  <resource-adaptor-entity-link>DiameterShServer</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 `DiameterShServerResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=DiameterShServer]`.

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 hardware's main concern is RAM memory and Java Heap size, the more the better. For instance, while the underlying JBoss Communications JAIN SLEE may run with 1GB of RAM, 8GB is needed to achieve performance higher than 800 new requests per second.

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

4.1.2. Software Prerequisites

The RA requires JBoss Communications JAIN SLEE properly set and Mobicents Diameter Multiplexer (MUX), which includes the stack, and JBoss Communications Diameter Base RA to be properly installed too.

4.2. JBoss Communications JAIN SLEE Diameter Sh-Server 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 2.6.1.FINAL.

```
[usr]$ svn co ?/2.6.1.FINAL slee-ra-diameter-sh-server-2.6.1.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-diameter-sh-server-2.6.1.FINAL
[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 Diameter Sh-Server 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 Diameter Sh-Server 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

5.1. Failover

The Diameter stack used by the JBoss Communications JAIN SLEE Diameter Sh-Server Resource Adaptor supports application session failover, with specific session state being replicated, thus only available for Application sessions. Failover of application activities is transparent to SLEE applications. This means that SLEE applications must be in charge of properly adapting its state machine to recover generic session on node failure.

5.2. Load Balancing

Currently, the only available balancing mechanism is provided by Diameter stack. It depends on [RFC 3588](http://tools.ietf.org/html/rfc3588) [http://tools.ietf.org/html/rfc3588] algorithm to select one peer from realm serving the desired application.

Appendix A. Revision History

Revision History

Revision 1.0

Tue Feb 09 2010

AlexandreMendonça

Creation of the JBoss Communications JAIN SLEE Diameter Sh-Server RA User Guide.

Index

F

feedback, viii

