

JBoss Communications JAIN SLEE Diameter Rf 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 Rf Resource Adaptor...	1
2. Resource Adaptor Type	3
2.1. Activities	3
2.2. Events	4
2.3. Activity Context Interface Factory	5
2.4. Resource Adaptor Interface	5
2.5. Restrictions	7
2.6. Sbb Code Examples	7
3. Resource Adaptor Implementation	9
3.1. Configuration	9
3.2. Default Resource Adaptor Entities	9
3.3. Traces and Alarms	10
3.3.1. Tracers	10
3.3.2. Alarms	10
4. Setup	11
4.1. Pre-Install Requirements and Prerequisites	11
4.1.1. Hardware Requirements	11
4.1.2. Software Prerequisites	11
4.2. JBoss Communications JAIN SLEE Diameter Rf Resource Adaptor Source Code..	11
4.2.1. Release Source Code Building	11
4.2.2. Development Trunk Source Building	12
4.3. Installing JBoss Communications JAIN SLEE Diameter Rf Resource Adaptor	12
4.4. Uninstalling JBoss Communications JAIN SLEE Diameter Rf Resource Adaptor...	12
5. Clustering	15
5.1. Failover	15
5.2. Load Balancing	15
A. Revision History	17
Index	19

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 Rf Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_DIAMETER_RF_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 Rf Resource Adaptor

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

Rf is one of two IMS reference point for charging. It covers offline charging - scenarios where charging information *does not* affect, in real-time, the service rendered.

Offline Charging occurs between two peers: the CTF (the online client, which issues request for accounting) and CDF (the server which performs accounting).

Two cases are currently distinguished for offline charging purposes:

- Event based charging;
- Session based charging.

For offline charging the Diameter Base Accounting defined in IETF [RFC 3588](http://tools.ietf.org/html/rfc3588) [http://tools.ietf.org/html/rfc3588] is used with additional AVPs defined. Also it reuses the FSM from Diameter Base Accounting.

Events are reused from Diameter Base Resource Adaptor and represent messages received by the Diameter stack. Different events types are specified for each Diameter request or answer. Events are fired either on client or server activities.

The Activities are defined by RA Type (and are an extension to Base Accounting Activities) 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 Rf Resource Adaptor Type is defined by Mobicents team as part of effort to standardize RA Types.

2.1. Activities

Diameter Rf Type 2.6.1.FINAL defines the following Activities:

`net.java.slee.resource.diameter.rf.RfClientSession`

This type of activity represents client side of Rf session. Accounting-Request (ACR) messages 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 proper `createRfClientSessionActivity` method of `net.java.slee.resource.diameter.rf.RfProvider`. It ends once underlying Base Accounting session ends.

State machines for client Rf are based on Base Accounting client sessions which can be found at [Section 8.2](http://tools.ietf.org/html/rfc3588#section-8.2) [http://tools.ietf.org/html/rfc3588#section-8.2] of Diameter Base Protocol RFC.

`net.java.slee.resource.diameter.rf.RfServerSession`

This type of activity represents server side of Rf session. Accounting-Request (ACR) are received in this Activity and respective Answers are sent from it.

This activity type is implicitly created by the Resource Adaptor upon reception of the Accounting-Request message. It ends once underlying Base Accounting session ends.

State machines for client Rf are based on Base Accounting client sessions which can be found at [Section 8.2](http://tools.ietf.org/html/rfc3588#section-8.2) [http://tools.ietf.org/html/rfc3588#section-8.2] of Diameter Base Protocol RFC.

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

```
public AccountingAnswer createRfAccountingAnswer();

public AccountingAnswer createRfAccountingAnswer(AccountingRequest acr);

public void sendAccountingAnswer(AccountingAnswer accountingAnswer)
    throws IOException, IllegalArgumentException;
```

```
public AccountingAnswer createRfAccountingAnswer();
```

This method creates an Accounting-Answer with the Acct-Application-Id set to 3.

```
public AccountingAnswer createRfAccountingAnswer(AccountingRequest acr);
```

This method creates an Accounting-Answer with some AVPs populated from the provided Accounting-Request.

The ACA will contain the AVPs specified in createRfAccountingAnswer() and the following AVPs from the Accounting-Request:

- Accounting-Record-Type
- Accounting-Record-Number

```
public void sendAccountingAnswer(AccountingAnswer accountingAnswer) throws IOException,  
IllegalArgumentException;
```

This method sends an Accounting Answer.

Rf Client Activity is defined as follows:

```
public void sendAccountingRequest(AccountingRequest accountingRequest)  
throws IOException, IllegalArgumentException;
```

```
public void sendAccountingRequest(AccountingRequest accountingRequest) throws  
IOException, IllegalArgumentException;
```

This method sends an Accounting Request.



Note

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

2.2. Events

Diameter Rf Resource Adaptor Type declares the Base Accounting Application specific events, ie, Accounting-Request/Answer.

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

Table 2.1. Events received on Rf Server Activity

Name	Vendor	Version	Class
net.java.slee.resource.diameter.base.events.AccountingRequest	java.net	0.8	net.java.slee.resource.diameter.base.events.AccountingRequest

Table 2.2. Events received on Rf Client Activity

Name	Vendor	Version	Class
net.java.slee.resource. diameter.base.events. AccountingAnswer	java.net	0.8	net.java.slee.resource. diameter.base.events. AccountingAnswer

**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 Rf Activity Context Interface Factory is defined as follows:

```
package net.java.slee.resource.diameter.rf;

import javax.slee.ActivityContextInterface;

public interface RfActivityContextInterfaceFactory {

    public ActivityContextInterface getActivityContextInterface(RfClientSession cSession);

    public ActivityContextInterface getActivityContextInterface(RfServerSession sSession);

}
```

2.4. Resource Adaptor Interface

The JBoss Communications Diameter Rf 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.rf;

import java.io.IOException;
```

```
import net.java.slee.resource.diameter.base.CreateActivityException;
import net.java.slee.resource.diameter.base.events.AccountingAnswer;
import net.java.slee.resource.diameter.base.events.AccountingRequest;
import net.java.slee.resource.diameter.base.events.avp.DiameterIdentity;

public interface RfProvider {

    public RfMessageFactory getRfMessageFactory();

    public RfClientSession createRfClientSessionActivity() throws CreateActivityException;

    public RfClientSession createRfClientSessionActivity(DiameterIdentity destinationHost,
        DiameterIdentity destinationRealm) throws CreateActivityException;

    public AccountingAnswer sendAccountingRequest(AccountingRequest accountingRequest)
        throws IllegalArgumentException, IOException;

    DiameterIdentity[] getConnectedPeers();

    int getPeerCount();
}
```

`public RfMessageFactory getRfMessageFactory();`

This method returns a message factory to be used to create concrete implementations of accounting messages.

`public RfClientSession createRfClientSessionActivity() throws CreateActivityException;`

This method creates a new activity to send and receive Diameter messages.

`public RfClientSession createRfClientSessionActivity(DiameterIdentity destinationHost, DiameterIdentity destinationRealm) throws CreateActivityException;`

This method creates a new activity to send and receive Diameter messages.

`public AccountingAnswer sendAccountingRequest(AccountingRequest accountingRequest) throws IllegalArgumentException, IOException;`

This method sends an Accounting Request.

`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

TODO

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
acctApplicationIds	List of supported Accounting Application Ids in form of {vendor}: {application-id}, separated by comma ','	java.lang.String	10415:3



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

The `DiameterRf` entity is also bound to Resource Adaptor Link Name `DiameterRf`, 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 Rf</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/JDiameterRfResourceAdaptor/java.net/0.8.1/acif
</activity-context-interface-factory-name>

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

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 Rf 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-rf-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-rf-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 Rf 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 Rf 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 Rf 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

Wed Feb 10 2010

AlexandreMendonça

Creation of the JBoss Communications JAIN SLEE Diameter Rf RA User Guide.

Index

F

feedback, viii

