JBoss Communications
JAIN SLEE Diameter Base
Resource Adaptor User Guide

by 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 Diameter Base Resource Adaptor**, or contact the authors.

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

This resource adaptor provides a Diameter API for JAIN SLEE applications, according to Diameter Base protocol, responsible for managing connection between peers and provide an Authentication, Authorization and Accounting (AAA) framework for applications such as network access or IP mobility, as defined by the protocol specs done by the IETF in *RFC 3588* [http://tools.ietf.org/html/rfc3588].

Events represent Diameter Base 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 activity.

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 Base Resource Adaptor Type is defined by Mobicents team as part of effort to standardize RA Types.

2.1. Activities

Diameter Base Type 2.6.1.FINAL defines the following Activities:

net.java.slee.resource.diameter.base.AuthClientSessionActivity

This type of activity represents client side of Base Authentication session. Abort-Session-Request (ASR) and Re-Auth-Request (RAR) messages are received in this Activity and respective Answers are sent from it. Session-Termination-Request (STR) 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 appropriate `createAuthenticationClientSessionActivity` method of `net.java.slee.resource.diameter.base.DiameterProvider`. It ends once underlying Base Authentication session ends.

State machine for client Base Authorization sessions can be found at *Section 8.1* [http://tools.ietf.org/html/rfc3588#section-8.1] of Diameter Base Protocol RFC.

net.java.slee.resource.diameter.base.AuthServerSessionActivity

This type of activity represents server side of Base Authentication session. Session-Termination-Request (STR) messages are received in this Activity and respective Answers are sent from it. Abort-Session-Request (ASR) and Re-Auth-Request (RAR) 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 appropriate `createAuthenticationServerSessionActivity` method of `net.java.slee.resource.diameter.base.DiameterProvider`. It ends once underlying Base Authentication session ends.

State machine for client Base Authorization sessions can be found at *Section 8.1* [http://tools.ietf.org/html/rfc3588#section-8.1] of Diameter Base Protocol RFC.

net.java.slee.resource.diameter.base.AccountingClientSessionActivity

This type of activity represents client side of Base Accounting 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 appropriate `createAccountingClientSessionActivity` method of `net.java.slee.resource.diameter.base.DiameterProvider`. It ends once underlying Base Accounting session ends.

State machine for client Base Accounting sessions can be found at *Section 8.2* [http://tools.ietf.org/html/rfc3588#section-8.2] of Diameter Base Protocol RFC.

net.java.slee.resource.diameter.base.AccountingServerSessionActivity
: This type of activity represents server side of Base Accounting session. Accounting-Request (ACR) messages are received in this Activity and respective Answers are sent from it.

  This activity type is created explicitly for incoming requests by Resource Adaptor. It ends once underlying Base Accounting session ends, either by timeout or proper answer delivery.

  State machine for server Base Accounting can be found at *Section 8.2* [http://tools.ietf.org/html/rfc3588#section-8.2] of Diameter Base Protocol RFC.

net.java.slee.resource.diameter.base.DiameterActvity
: This type of activity represents a generic Diameter session and it's the super type for all other Diameter Activities. It can be used as an extension point for Diameter Applications.

  This activity type is created explicitly for unknown (not defined in other Activities) incoming requests by Resource Adaptor or by calling the appropriate `createActivity` method of `net.java.slee.resource.diameter.base.DiameterProvider`. It ends once underlying session ends, either by timeout or proper answer delivery.

All activities define methods required to properly function and expose necessary information to JAIN SLEE services. Common part for all Diameter Activities is defined as follows:

```
DiameterMessageFactory getDiameterMessageFactory();

DiameterAvpFactory getDiameterAvpFactory();

void sendMessage(DiameterMessage message) throws IOException;

String getSessionId();

void endActivity();
```

DiameterMessageFactory getDiameterMessageFactory();
: Returns message factory capable of creating Diameter Base or other (extension) messages.

DiameterAvpFactory getDiameterAvpFactory();
: Returns AVP factory capable of creating Diameter Base or other (extension) AVPs.

void sendMessage(DiameterMessage message) throws IOException;
: Provides means to send generic Diameter messages.

String getSessionId();
Returns Session ID of underlying Diameter session.

void endActivity();
Terminates underlying session.

Common part for Authentication Activities is defined as follows:

```
AuthSessionState getSessionState();
```

AuthSessionState getSessionState();
Return current Auth session state. It can have values as follows:

- IDLE
- PENDING
- OPEN
- DISCONNECTED

DISCONNECTED value implies that activity is ending.

Client part for Authentication Activities is defined as follows:

```
AbortSessionAnswer createAbortSessionAnswer();

AbortSessionAnswer createAbortSessionAnswer(AbortSessionRequest abortSessionRequest);

void sendAbortSessionAnswer(AbortSessionAnswer abortSessionAnswer)
    throws IOException, IllegalArgumentException;

ReAuthAnswer createReAuthAnswer();

ReAuthAnswer createReAuthAnswer(ReAuthRequest reAuthRequest);

void sendReAuthAnswer(ReAuthAnswer reAuthAnswer)
    throws IOException, IllegalArgumentException;

SessionTerminationRequest createSessionTerminationRequest(
    TerminationCauseType terminationCause);

void sendSessionTerminationRequest(
```

```
        SessionTerminationRequest sessionTerminationRequest)
            throws IOException, IllegalArgumentException;
```

AbortSessionAnswer createAbortSessionAnswer();

Create an empty Abort-Session-Answer with the Auth-Application-Id set to 0.

AbortSessionAnswer createAbortSessionAnswer(AbortSessionRequest abortSessionRequest);

Create an Abort-Session-Answer with some AVPs populated from the provided Abort-Session-Request.

void sendAbortSessionAnswer(AbortSessionAnswer abortSessionAnswer) throws IOException, IllegalArgumentException;

Send an Abort-Session-Answer.

ReAuthAnswer createReAuthAnswer();

Create an empty Re-Auth-Answer with the Auth-Application-Id set to 0.

ReAuthAnswer createReAuthAnswer(ReAuthRequest reAuthRequest);

Create an Re-Auth-Answer with some AVPs populated from the provided Re-Auth-Request.

void sendReAuthAnswer(ReAuthAnswer reAuthAnswer) throws IOException, IllegalArgumentException;

Send an Re-Auth-Answer.

SessionTerminationRequest createSessionTerminationRequest(TerminationCauseType terminationCause);

Create an Session-Termination-Request message populated with the following AVPs:

- Termination-Cause: as per terminationCause parameter
- Auth-Application-Id: the value 0 as specified by RFC3588

void sendSessionTerminationRequest(SessionTerminationRequest sessionTerminationRequest) throws IOException, IllegalArgumentException;

Send an Session Termination Request. An event containing the answer will be fired on this activity.

Server part for Authentication Activities is defined as follows:

```
    AbortSessionRequest createAbortSessionRequest();

    void sendAbortSessionRequest(AbortSessionRequest request) throws IOException;

    ReAuthRequest createReAuthRequest(ReAuthRequestType reAuthRequestType);

    void sendReAuthRequest(ReAuthRequest request) throws IOException;
```

```
SessionTerminationAnswer createSessionTerminationAnswer();

SessionTerminationAnswer createSessionTerminationAnswer(
    SessionTerminationRequest sessionTerminationRequest);

void sendSessionTerminationAnswer(SessionTerminationAnswer request)
    throws IOException;
```

AbortSessionRequest createAbortSessionRequest();
Create an Abort-Session-Request message populated with the following AVPs:

- Auth-Application-Id: the value 0 as specified by RFC3588

void sendAbortSessionRequest(AbortSessionRequest request) throws IOException;
Send session abort session request to client

ReAuthRequest createReAuthRequest(ReAuthRequestType reAuthRequestType);
Create an Re-Auth-Request message populated with the following AVPs:

- Auth-Application-Id: the value 0 as specified by RFC3588
- Re-Auth-Request-Type: as per reAuthRequestType parameter

void sendReAuthRequest(ReAuthRequest request) throws IOException;
Send Re-Auth-Request.

SessionTerminationAnswer createSessionTerminationAnswer();
Create an Session-Termination-Answer with the Auth-Application-Id set to 0.

SessionTerminationAnswer createSessionTerminationAnswer(SessionTerminationRequest sessionTerminationRequest);
Create an Session-Termination-Answer with some AVPs populated from the provided Session-Termination-Request.

void sendSessionTerminationAnswer(SessionTerminationAnswer request) throws IOException;
Send session termination answer to client.

Common part for Accounting Activities is defined as follows:

```
AccountingSessionState getAccountingSessionState();
```

AccountingSessionState getAccountingSessionState();
Returns current Accounting session state of underlying session. Valid values are:

- Idle
- PendingS
- PendingE
- PendingB
- Open
- PendingI
- PendingL

Client part for Accounting Activities is defined as follows:

```
AccountingRequest createAccountingRequest(AccountingRecordType accountingRecordType);

void sendAccountRequest(AccountingRequest accountingRequest)
    throws IOException, IllegalArgumentException;
```

AccountingRequest createAccountingRequest(AccountingRecordType accountingRecordType);
Create an Accounting-Request message populated with the following AVPs:

- Accounting-Record-Type: as per accountingRecordType parameter
- Acct-Application-Id: the value 3 as specified by RFC3588

void sendAccountRequest(AccountingRequest accountingRequest) throws IOException, IllegalArgumentException;
Send an Accounting Request. An event containing the answer or timeout will be fired on this activity.

Server part for Accounting Activities is defined as follows:

```
AccountingAnswer createAccountingAnswer();

AccountingAnswer createAccountingAnswer(AccountingRequest acr);

void sendAccountingAnswer(AccountingAnswer aca)
    throws IOException, IllegalArgumentException;
```

AccountingAnswer createAccountingAnswer();

Create an Accounting-Answer with the Acct-Application-Id set to 3.

AccountingAnswer createAccountingAnswer(AccountingRequest acr);

Create an Accounting-Answer with some AVPs populated from the provided Accounting-Request. The ACA will contain the AVPs specified in createAccountingAnswer() and the following AVPs from the Accounting-Request:

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

void sendAccountingAnswer(AccountingAnswer aca) throws IOException, IllegalArgumentException;

Send an Accounting Answer.

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 this section.

2.2. Events

Diameter Base Resource Adaptor Type declares all the Diameter Base messages but the Peer Management related messages are not usable by Services, ie, are not fired.

The following tables shows which events are fired on each activity and the ones which aren't fired on any.

Table 2.1. Events received on Authorization Server Activity

Name	Vendor	Version	Class
net.java.slee.resource.diameter.base.events.SessionTerminationRequest	java.net	0.8	net.java.slee.resource.diameter.base.events.SessionTerminationRequest
net.java.slee.resource.diameter.base.events.AbortSessionAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.AbortSessionAnswer
net.java.slee.resource.diameter.base.events.ReAuthAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.ReAuthAnswer
net.java.slee.resource.diameter.base.events.ErrorAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.ErrorAnswer

Table 2.2. Events received on Authorization Client Activity

Name	Vendor	Version	Class
net.java.slee.resource.diameter.base.events.AbortSessionRequest	java.net	0.8	net.java.slee.resource.diameter.base.events.AbortSessionRequest
net.java.slee.resource.diameter.base.events.SessionTerminationAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.SessionTerminationAnswer
net.java.slee.resource.diameter.base.events.ReAuthRequest	java.net	0.8	net.java.slee.resource.diameter.base.events.ReAuthRequest
net.java.slee.resource.diameter.base.events.ErrorAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.ErrorAnswer

Table 2.3. Events received on Accounting 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 2.4. Events received on Accounting 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
net.java.slee.resource.diameter.base.events.ErrorAnswer	java.net	0.8	net.java.slee.resource.diameter.base.events.ErrorAnswer

Table 2.5. Events received on Generic Diameter Activity

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

Table 2.6. Events defined but not received on any Activity (for possible future usage)

Name	Vendor	Version	Class
net.java.slee.resource. diameter.base.events. CapabilitiesExchangeRequest	java.net	0.8	net.java.slee.resource. diameter.base.events. CapabilitiesExchangeRequest
net.java.slee.resource. diameter.base.events. CapabilitiesExchangeAnswer	java.net	0.8	net.java.slee.resource. diameter.base.events. CapabilitiesExchangeAnswer
net.java.slee.resource. diameter.base.events. DeviceWatchdogRequest	java.net	0.8	net.java.slee.resource. diameter.base.events. DeviceWatchdogRequest
net.java.slee.resource. diameter.base.events. DeviceWatchdogAnswer	java.net	0.8	net.java.slee.resource. diameter.base.events. DeviceWatchdogAnswer
net.java.slee.resource. diameter.base.events. DisconnectPeerRequest	java.net	0.8	net.java.slee.resource. diameter.base.events. DisconnectPeerRequest
net.java.slee.resource. diameter.base.events. DisconnectPeerAnswer	java.net	0.8	net.java.slee.resource. diameter.base.events. DisconnectPeerAnswer

Important

Spaces where 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 Base Activity Context Interface Factory is defined as follows:

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

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

public interface DiameterActivityContextInterfaceFactory {
```

```
    public ActivityContextInterface
        getActivityContextInterface(DiameterActivity activity)
            throws UnrecognizedActivityException, FactoryException;

    public ActivityContextInterface
        getActivityContextInterface(AccountingClientSessionActivity activity)
            throws UnrecognizedActivityException;

    public ActivityContextInterface
        getActivityContextInterface(AccountingServerSessionActivity activity)
            throws UnrecognizedActivityException;
}
```

2.4. Resource Adaptor Interface

The JBoss Communications Diameter Base 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.base;

import net.java.slee.resource.diameter.base.events.DiameterMessage;
import net.java.slee.resource.diameter.base.events.avp.DiameterIdentity;

import java.io.IOException;

public interface DiameterProvider {

    public DiameterMessageFactory
        getDiameterMessageFactory();

    public DiameterAvpFactory getDiameterAvpFactory();

    public DiameterActivity createActivity() throws CreateActivityException;

    public DiameterActivity createActivity(DiameterIdentity destinationHost,
        DiameterIdentity destinationRealm) throws CreateActivityException;

    public AccountingClientSessionActivity createAccountingClientSessionActivity()
        throws CreateActivityException;
```

```
    public AccountingClientSessionActivity createAccountingClientSessionActivity(
        DiameterIdentity destinationHost, DiameterIdentity destinationRealm)
            throws CreateActivityException ;

    public AuthClientSessionActivity createAuthenticationClientSessionActivity()
        throws CreateActivityException;

    public AuthClientSessionActivity createAuthenticationClientSessionActivity(
        DiameterIdentity destinationHost, DiameterIdentity destinationRealm)
            throws CreateActivityException;

    public DiameterMessage sendSyncRequest(DiameterMessage message)
        throws IOException;

    public DiameterIdentity[] getConnectedPeers();

    public int getPeerCount();
}
```

public DiameterMessageFactory getDiameterMessageFactory();

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

public DiameterAvpFactory getDiameterAvpFactory();

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

public DiameterActivity createActivity() throws CreateActivityException;

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

public DiameterActivity createActivity(DiameterIdentity destinationHost, DiameterIdentity destinationRealm) throws CreateActivityException;

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

public AccountingClientSessionActivity createAccountingClientSessionActivity() throws CreateActivityException;

This method create a new activity to send Diameter Accounting request messages.

public AccountingClientSessionActivity createAccountingClientSessionActivity(DiameterIdentity destinationHost, DiameterIdentity destinationRealm) throws CreateActivityException ;

This method create a new activity to send Diameter Accounting request messages.

public AuthClientSessionActivity createAuthenticationClientSessionActivity() throws CreateActivityException;

This method create a new activity to send Diameter Authorization request messages.

public AuthClientSessionActivity createAuthenticationClientSessionActivity(DiameterIdentity destinationHost, DiameterIdentity destinationRealm)throws CreateActivityException;

This method create a new activity to send Diameter Authorization request messages.

public DiameterMessage sendSyncRequest(DiameterMessage message) throws IOException;

Synchronously send a Diameter request and block until a response is received.

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 Client-Side Example that creates and sends an Accounting-Request and receives an Accounting-Answer.

```
/* Method for creating and sending ACR with pre-defined values. */
private void sendAccountingRequest() {
   try {
   AccountingClientSessionActivity activity = provider.createAccountingClientSessionActivity();

      List<DiameterAvp> avps = new ArrayList<DiameterAvp>();

      avps.add(avpFactory.createAvp(DiameterAvpCodes.SESSION_ID,
         activity.getSessionId().getBytes()));

     DiameterAvp avpVendorId = avpFactory.createAvp(DiameterAvpCodes.VENDOR_ID, 193);
      DiameterAvp avpAcctApplicationId = avpFactory.createAvp(
         DiameterAvpCodes.ACCT_APPLICATION_ID, 19302);

      avps.add(avpFactory.createAvp(
         DiameterAvpCodes.VENDOR_SPECIFIC_APPLICATION_ID,
            new DiameterAvp[] { avpVendorId, avpAcctApplicationId }));

      avps.add(avpFactory.createAvp(DiameterAvpCodes.ORIGIN_HOST,
         this.originIP.getBytes()));
      avps.add(avpFactory.createAvp(DiameterAvpCodes.ORIGIN_REALM,
         this.originRealm.getBytes()));
```

```
        avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_HOST,
            (this.destinationIP + ":" + this.destinationPort).getBytes()));
        avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_REALM,
            this.destinationRealm.getBytes()));

        // Subscription ID
        DiameterAvp subscriptionIdType = avpFactory.createAvp(193, 555, 0);
        DiameterAvp subscriptionIdData = avpFactory.createAvp(193, 554, "00001000");
        avps.add(avpFactory.createAvp(193, 553, new DiameterAvp[] {subscriptionIdType,
            subscriptionIdData}));

        // Requested Service Unit
        DiameterAvp unitType = avpFactory.createAvp(193, 611, 2);
        DiameterAvp valueDigits = avpFactory.createAvp(193, 617, 10L);
    DiameterAvp unitValue = avpFactory.createAvp(193, 612, new DiameterAvp[] {valueDigits});
        avps.add(avpFactory.createAvp(193, 606, new DiameterAvp[] {unitType, unitValue}));

        // Record Number and Type
        avps.add(avpFactory.createAvp(
            DiameterAvpCodes.ACCOUNTING_RECORD_NUMBER, 0));
        avps.add(avpFactory.createAvp(
            DiameterAvpCodes.ACCOUNTING_RECORD_TYPE, 1));

        // Requested action
        avps.add(avpFactory.createAvp(193, 615, 0));

        // Service Parameter Type
        DiameterAvp serviceParameterType = avpFactory.createAvp(193, 608, 0);
        DiameterAvp serviceParameterValue = avpFactory.createAvp(193, 609, "510");
        avps.add(avpFactory.createAvp(193, 607, new DiameterAvp[] {serviceParameterType,
            serviceParameterValue}));

        // Service Parameter Type
        DiameterAvp serviceParameterType2 = avpFactory.createAvp(193, 608, 14);
        DiameterAvp serviceParameterValue2 = avpFactory.createAvp(193, 609, "20");
        avps.add(avpFactory.createAvp(193, 607, new DiameterAvp[] {serviceParameterType2,
            serviceParameterValue2}));

        DiameterAvp[] avpArray = new DiameterAvp[avps.size()];
        avpArray = avps.toArray(avpArray);

        if (tracer.isInfoEnabled())
            tracer.info("Creating Custom Message...");
```

```
        AccountingRequest acr = messageFactory.createAccountingRequest(avpArray);

        if (tracer.isFineEnabled()) {
            tracer.fine("Created Custom Message[" + acr + "]");
            tracer.fine("Sending Custom Message...");
        }

        ActivityContextInterface aci = acif.getActivityContextInterface(activity);

        activity.sendAccountRequest(acr);

        aci.attach(sbbContext.getSbbLocalObject());

        if (tracer.isInfoEnabled()) {
            tracer.info("Sent Custom Message[" + acr + "]");
        }
    }
    catch (Exception e) {
        tracer.severe("", e);
    }
}

...

/* Method for handling ACA messages. Just print the Result-Code AVP. */
public void onAccountingAnswer(AccountingAnswer aca, ActivityContextInterface aci) {
    if (tracer.isInfoEnabled()) {
        tracer.info("Accounting-Answer received. Result-Code[" + aca.getResultCode() + "].");
    }
}
```

Simple Server-Side Example for receiving Accounting-Request and proxy it to it's destination, if acting as proxy, or answer with Accounting-Answer, if in Server mode.

```
/* Method for handling ACR messages and either proxy or answer them. */
public void onAccountingRequest(AccountingRequest acr, ActivityContextInterface aci) {
    long start = System.currentTimeMillis();
    if (tracer.isInfoEnabled())
        tracer.info("Accounting-Request received. [" + acr + "]");

    boolean actAsProxy = false;
```

```
try {
    // Are we gonna act as a proxy?
    if (actAsProxy) {
        // In here we act as a "proxy". Just for testing we take the original message,
        // replace the Origin/Destination Host/Realm AVPs and send it to the emulator.

        boolean hasDestinationHost = false;
        boolean hasDestinationRealm = false;

        List<DiameterAvp> avps = new ArrayList<DiameterAvp>();

        for (DiameterAvp avp : acr.getAvps()) {
            switch (avp.getCode()) {
            case DiameterAvpCodes.ORIGIN_HOST:
                avps.add(avpFactory.createAvp(DiameterAvpCodes.ORIGIN_HOST, ("aaa://" +
                    originIP + ":" + originPort).getBytes()));
                break;
            case DiameterAvpCodes.ORIGIN_REALM:
                avps.add(avpFactory.createAvp(DiameterAvpCodes.ORIGIN_REALM,
                    originRealm.getBytes()));
                break;
            case DiameterAvpCodes.DESTINATION_HOST:
                avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_HOST,
                    ("aaa://" + destinationIP + ":" + destinationPort).getBytes()));
                hasDestinationHost = true;
                break;
            case DiameterAvpCodes.DESTINATION_REALM:
                avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_REALM,
                    destinationRealm.getBytes()));
                hasDestinationRealm = true;
                break;
            default:
                avps.add(avp);
            }
        }

        if (!hasDestinationHost)
            avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_HOST,
                "127.0.0.1".getBytes()));

        if (!hasDestinationRealm)
            avps.add(avpFactory.createAvp(DiameterAvpCodes.DESTINATION_REALM,
                "mobicents.org".getBytes()));
```

```
            if (tracer.isInfoEnabled())
                tracer.info("AVPs ==> " + avps);

            DiameterAvp[] avpArray = new DiameterAvp[avps.size()];
            avpArray = avps.toArray(avpArray);
            if (tracer.isInfoEnabled())
                tracer.info("Creating Custom Message...");
            DiameterMessage ms = messageFactory.createAccountingRequest(avpArray);
            if (tracer.isInfoEnabled()) {
                tracer.info("Created Custom Message[" + ms + "]");

                tracer.info("Sending Custom Message...");
            }
            provider.createActivity().sendMessage(ms);
            if (tracer.isInfoEnabled()) {
                tracer.info("Sent Custom Message[" + ms + "]");
            }
        }
        else {
            // In here we act as a server and just say it's SUCCESS.

            if (aci.getActivity() instanceof AccountingServerSessionActivity) {
                AccountingServerSessionActivity assa =
                    (AccountingServerSessionActivity) aci.getActivity();

                AccountingAnswer ans = assa.createAccountingAnswer(acr);
                ans.setResultCode(2001); // 2001 = SUCCESS

                if (tracer.isInfoEnabled()) {
                    tracer.info("Sending Accounting-Answer [" + ans + "]");
                }

                assa.sendAccountingAnswer(ans);
                if (tracer.isInfoEnabled()) {
                    tracer.info("Accounting-Answer sent.");
                }
            }
        }
    }
    catch (Exception e) {
        tracer.severe("", e);
    }

    long end = System.currentTimeMillis();
```

```
    if (tracer.isInfoEnabled()) {
        tracer.info("Accounting-Request proccessed. [" + (end - start) + "ms]");
    }
}
```

Resource Adaptor Implementation

This RA uses the JBoss Communications Diameter Stack, an improvement over *jDiameter Stack* [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	0:4, 193:19301
acctApplicationIds	List of supported Accounting Application Ids in form of {vendor}:{application-id}, separated by comma ','	java.lang.String	0:3, 193:19302

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 `DiameterBaseResourceAdaptor`. The `DiameterBaseResourceAdaptor` entity uses the default Resource Adaptor configuration, specified in *Section 3.1, "Configuration"*.

The `DiameterBaseResourceAdaptor` entity is also bound to Resource Adaptor Link Name `DiameterBaseResourceAdaptor`, 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 Base</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/diameter-base-ra/acif
    </activity-context-interface-factory-name>

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

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, to be properly installed too.

4.2. JBoss Communications JAIN SLEE Diameter Base 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-base-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-base-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 Base 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 Base 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 Base 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] algorithm to select one peer from realm serving the desired application.

Appendix A. Revision History

Revision History
Revision 1.0 Mon Feb 08 2010 AlexandreMendonça
Creation of the JBoss Communications JAIN SLEE Diameter Base RA User Guide.

Index

F