JBoss Communications JAIN SLEE MGCP Resource Adaptor User Guide

by Amit Bhayani

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 MGCP Resource Adaptor 1
2. Resource Adaptor Type 3
2.1. Activities 3
2.2. Events 4
2.3. Activity Context Interface Factory 7
2.4. Resource Adaptor Interface 8
2.5. Restrictions 9
2.6. Sbb Code Examples 9
3. Resource Adaptor Implementation 19
3.1. Configuration 19
3.2. Default Resource Adaptor Entities 19
3.3. Traces and Alarms 20
3.3.1. Tracers 20
3.3.2. Alarms 20
4. Setup 21
4.1. Pre-Install Requirements and Prerequisites 21
4.1.1. Hardware Requirements 21
4.1.2. Software Prerequisites 21
4.2. JBoss Communications JAIN SLEE MGCP Resource Adaptor Source Code 21
4.2.1. Release Source Code Building 21
4.2.2. Development Trunk Source Building 22
4.3. Installing JBoss Communications JAIN SLEE MGCP Resource Adaptor 22
4.4. Uninstalling JBoss Communications JAIN SLEE MGCP Resource Adaptor 22
5. Clustering 25
A. Revision History 27
Index 29

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 MGCP Resource Adaptor**, or contact the authors.

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

This resource adaptor provides a MGCP API for JAIN SLEE applications, adapting the JAIN MGCP specification - JSR 23. JAIN MGCP is a Java specification for the Media Gateway Control Protocol, as defined by the protocol specs done by the IETF RFC3435.

MGCP assumes a connection model where the basic constructs are endpoints and connections. Endpoints are sources and/or sinks of data and can be physical or virtual.

Connections may be either point to point or multipoint. A point to point connection is an association between two endpoints with the purpose of transmitting data between these endpoints. Once this association is established for both endpoints, data transfer between these endpoints can take place. A multipoint connection is established by connecting the endpoint to a multipoint session.

Both endpoints and connection constructs are available to support all type of MGCP Applications.

Examples of Application using Connections are

Transmission of audio packets using RTP and UDP over an IP network.

Transmission of packets over an internal connection, for example the TDM backplane or the interconnection bus of a gateway. This is used, in particular, for "hairpin" connections, connections that terminate in a gateway but are immediately rerouted over the telephone network.

Examples of Applications using Endpoints are

An example of a virtual endpoint is an audio source in an audio-content server.

An interface on a gateway that terminates a trunk connected to a PSTN switch (e.g., Class 5, Class 4, etc.). A gateway that terminates trunks is called a trunking gateway.

An interface on a gateway that terminates an analog POTS connection to a phone, key system, PBX, etc. A gateway that terminates residential POTS lines (to phones) is called a residential gateway.

Events represent MGCP Request/Response received by the MGCP stack, or failure use cases such as timeouts.

The Activities are the MGCP Connections and Endpoints, which applications in the SLEE may use to send MGCP Requests and Responses, and to receive the events related with incoming Request/Response.

Resource Adaptor Type

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

2.1. Activities

An MGCP activity object represents a set of related events in an MGCP resource. This Ra Type defines the following Activity objects:

MgcpConnectionActivity

The set of MGCP events regarding the managing (creation, modification, destruction and state audit) of a MGCP connection is represented by this activity object. This activity ends implicitly when a DeleteConnection (MGCP server role) or DeleteConnectionResponse (MGCP Call Agent role) event is received. MGCP events related to Signal generation or Event detection request for specific connection are also represented by this Activity. Class name is `net.java.slee.resource.mgcp.MgcpConnectionActivity`

New `MgcpConnectionActivity` Activity objects are created by calling `JainMgcpProvider.getConnectionActivity(int transactionHandle, EndpointIdentifier endpointIdentifier)` or `JainMgcpProvider.getConnectionActivity(ConnectionIdentifier connectionIdentifier, EndpointIdentifier endpointIdentifier)`

`JainMgcpProvider.getConnectionActivity(int transactionHandle, EndpointIdentifier endpointIdentifier)` is mostly used by Application behaving as Media Gateway Controller or MGCP call Agent Role. The above method is called to create a new `MgcpConnectionActivity` for unknown `ConnectionIdentifier` , to be used when sending `CreateConnection` events and receive further related messages from a MGCP Server.

`MgcpConnectionActivity` Activity objects are created automatically when the resource adaptor receives an incoming MGCP request (MGCP server role).

MgcpEndpointActivity

The set of MGCP events regarding a MGCP endpoint, such as event detecting/signal generation or configuration/state audit, is represented by this activity object. Since a MGCP endpoint doesn't have a lifecyle this activity does not end implicitly on MGCP events. Class name is `net.java.slee.resource.mgcp.MgcpConnectionActivity`

New `MgcpEndpointActivity` Activity objects is created by calling `JainMgcpProvider.public MgcpEndpointActivity getEndpointActivity(EndpointIdentifier endpointIdentifier);`

`MgcpEndpointActivity` Activity objects are created automatically when the resource adaptor receives an incoming MGCP request on an Endpoint (MGCP server role).

2.2. Events

Events represent MGCP Requests or Responses received by the MGCP stack (incoming requests and responses) and Timer expiry. Each MGCP Requests is fired as different event types. Each MGCP Response is fired as different event types. Events are fired on `MgcpConnectionActivity` or `MgcpEndpointActivity` activities. There are several event types defined. Following is the table that describes event-type (name, vendor and version), event-class and whether its fired on MgcpConnectionActivity or MgcpEndpointActivity or both.

Important

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

Important

For proper render of this table prefixes, for entries on some columns are ommited. For prefix values, for each column, please see list below:

Name
net.java.slee.resource.mgcp.

Event Class
jain.protocol.ip.mgcp.message.

Version for all defined events is 1.0.

Vendor for all defined events is net.java.

Table 2.1. Events fired by MGCP

Name	Event Class	Conn	Enp	Comments
CREATE_ CONNECTION	Create Connection	Yes	No	Received by application acting as MGCP Server.
CREATE_ CONNECTION_ RESPONSE	Create Connection Respone	Yes	No	Received by application acting as MGCP Call Agent
MODIFY_ CONNECTION	Modify Connection	Yes	No	Received by application acting as MGCP Server
MODIFY_ CONNECTION_ RESPONSE	Modify Connection Response	Yes	No	Received by application acting as MGCP Call Agent

Name	Event Class	Conn	Enp	Comments
DELETE_ CONNECTION	Delete Connection	Yes	Yes	Received by application acting as MGCP Server
DELETE_ CONNECTION_ RESPONSE	Delete Connection Response	Yes	Yes	Received by application acting as MGCP Call Agent
AUDIT_ CONNECTION	Audit Connection	Yes	No	Received by application acting as MGCP Server
AUDIT_ CONNECTION_ RESPONSE	Audit Connection Response	Yes	No	Received by application acting as MGCP Call Agent
NOTIFICATION_ REQUEST	Notification Request	Yes	Yes	Received by application acting as MGCP Server. Whether this event will be fired on MgcpConnectionActivity or MgcpEndpointActivity Activity is decided by RA depending on if EventName (Request/Signal) has ConnectionIdentifier included or not. If Request/Signal has list of EventName's with few having ConnectionIdentifier set and few without ConnectionIdentifier, then this event will be fired on both the Activities. However please note that new Activity Object/s will not be created unless initial-event="True" is specified in sbb-jar.xml for this event-type.
NOTIFICATION_ REQUEST_ RESPONSE	Notification Request Response	Yes	Yes	Received by application acting as MGCP Call Agent. If EventName (Signal/

Name	Event Class	Conn	Enp	Comments
				Request) in original NotificationRequest fired by this Application had ConnectionIdentifier set, then this event will be fired on MgcpConnectionActivity. If Request/Signal has list of EventName's with few having ConnectionIdentifier set and few without ConnectionIdentifier, then this event will be fired on both the Activities. However please note that event's will be fired only on existing Activty Object and no new Activity Object will be created unless initial-event="True" is specified in sbb-jar.xml.
NOTIFY	Notify	Yes	Yes	Received by application acting as MGCP Call Agent. Whether this event will be fired on MgcpConnectionActivity or MgcpEndpointActivity Activity is decided by RA depending on if event is detected on connection or endpoint (ConnectionIdentifier included or not)
NOTIFY_ RESPONSE	Notify Response	Yes	Yes	Received by application acting as MGCP Server. Whether this event will be fired on MgcpConnectionActivity or MgcpEndpointActivity

Name	Event Class	Conn	Enp	Comments
				Activity is decided by RA depending on if original Notify event is detected on connection or endpoint (ConnectionIdentifier included or not)
TRANSACTION_ TIMEOUT	event. Transaction Timeout	Yes	Yes	Received by application acting as MGCP Server or MGCP Call Agent. Basically this is fired when Application sends the MGCP Request but there is no response. The activity ends automatically.
AUDIT_ ENDPOINT	Audit Endpoint	No	Yes	Received by application acting as MGCP Server.
AUDIT_ ENDPOINT_ RESPONSE	Audit Endpoint Response	No	Yes	Received by application acting as MGCP Call Agent.
ENDPOINT_ CONFIGURATION	Endpoint Configuration	No	Yes	Received by application acting as MGCP Server.
ENDPOINT_ CONFIGURATION_ RESPONSE	Endpoint Configuration Response	No	Yes	Received by application acting as MGCP Call Agent.
RESTART_ IN_ PROGRESS	Restart In Progress	No	Yes	Received by application acting as MGCP Call Agent.
RESTART_ IN_ PROGRESS_ RESPONSE	Restart In Progress Response	No	Yes	Received by application acting as MGCP Server.

2.3. Activity Context Interface Factory

The interface of the JAIN MGCP resource adaptor type specific Activity Context Interface Factory is defined as follows:

```
package net.java.slee.resource.mgcp;

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


public interface MgcpActivityContextInterfaceFactory {


    public ActivityContextInterface getActivityContextInterface(MgcpEndpointActivity activity)
        throws NullPointerException, UnrecognizedActivityException, FactoryException;


    public ActivityContextInterface getActivityContextInterface(MgcpConnectionActivity activity)
        throws NullPointerException, UnrecognizedActivityException, FactoryException;

}
```

2.4. Resource Adaptor Interface

The JAIN MGCP Resource Adaptor SBB Interface provides SBBs with access to the JAIN MGCP objects required for creating and sending Request/Response. It is defined as follows:

```
 package net.java.slee.resource.mgcp;

import jain.protocol.ip.mgcp.message.CreateConnection;
import jain.protocol.ip.mgcp.message.NotificationRequest;
import jain.protocol.ip.mgcp.message.parms.CallIdentifier;
import jain.protocol.ip.mgcp.message.parms.ConnectionIdentifier;
import jain.protocol.ip.mgcp.message.parms.EndpointIdentifier;
import jain.protocol.ip.mgcp.message.parms.RequestIdentifier;

import java.util.List;


public interface JainMgcpProvider extends jain.protocol.ip.mgcp.JainMgcpProvider {

 public MgcpConnectionActivity getConnectionActivity(ConnectionIdentifier connectionIdentifier,
 EndpointIdentifier endpointIdentifier);
```

```
 public MgcpConnectionActivity getConnectionActivity(int transactionHandle, EndpointIdentifier endpointIdentifier);

      public List MgcpConnectionActivity getConnectionActivities(EndpointIdentifier endpointIdentifier);

 public MgcpEndpointActivity getEndpointActivity(EndpointIdentifier endpointIdentifier);

 public int getUniqueTransactionHandler();

 public CallIdentifier getUniqueCallIdentifier();

 public RequestIdentifier getUniqueRequestIdentifier();

}
```

2.5. Restrictions

The resource adaptor implementation should prevent SBBs from adding themselves as MGCP listeners, or changing the MGCP network configuration. Any attempt to do so should be rejected by throwing a SecurityException.

2.6. Sbb Code Examples

The following code shows how MGCP Call Agent can send CreateConnection request to MGCP Server

```
public abstract class IVRSbb implements Sbb {

 public final static String ENDPOINT_NAME = "/mobicents/media/IVR/$";


 //SIP Invite received. Send CRCX to MGCP Server
 public void onCallCreated(RequestEvent evt, ActivityContextInterface aci) {
  Request request = evt.getRequest();

  FromHeader from = (FromHeader) request.getHeader(FromHeader.NAME);
  ToHeader to = (ToHeader) request.getHeader(ToHeader.NAME);
```

```
 ActivityContextInterface daci = null;
 try {
  Dialog dialog = provider.getNewDialog(evt.getServerTransaction());
  dialog.terminateOnBye(true);
  daci = acif.getActivityContextInterface((DialogActivity) dialog);
  daci.attach(sbbContext.getSbbLocalObject());
 } catch (Exception e) {
  logger.severe("Error during dialog creation", e);
  respond(evt, Response.SERVER_INTERNAL_ERROR);
  return;
 }

 CallIdentifier callID = mgcpProvider.getUniqueCallIdentifier();
 this.setCallIdentifier(callID.toString());
        EndpointIdentifier endpointID = new EndpointIdentifier(ENDPOINT_NAME,
JBOSS_BIND_ADDRESS + ":" + MGCP_PEER_PORT);

    CreateConnection createConnection = new CreateConnection(this, callID, endpointID,
ConnectionMode.SendRecv);

 try {
  String sdp = new String(evt.getRequest().getRawContent());
  createConnection.setRemoteConnectionDescriptor(new ConnectionDescriptor(sdp));
 } catch (ConflictingParameterException e) {
  // should never happen
 }

 int txID = mgcpProvider.getUniqueTransactionHandler();
 createConnection.setTransactionHandle(txID);

 MgcpConnectionActivity connectionActivity = null;
 try {
  connectionActivity = mgcpProvider.getConnectionActivity(txID, endpointID);
  ActivityContextInterface epnAci = mgcpAcif.getActivityContextInterface(connectionActivity);
  epnAci.attach(sbbContext.getSbbLocalObject());
 } catch (FactoryException ex) {
  ex.printStackTrace();
 } catch (NullPointerException ex) {
  ex.printStackTrace();
 } catch (UnrecognizedActivityException ex) {
  ex.printStackTrace();
 }

 mgcpProvider.sendMgcpEvents(new JainMgcpEvent[] { createConnection });
```

```
}

//Received CRCX Response. Now Send RQNT for playing media file
    public void onCreateConnectionResponse(CreateConnectionResponse event,
ActivityContextInterface aci) {

 ServerTransaction txn = getServerTransaction();
 Request request = txn.getRequest();

 ReturnCode status = event.getReturnCode();

 switch (status.getValue()) {
 case ReturnCode.TRANSACTION_EXECUTED_NORMALLY:

  this.setEndpointName(event.getSpecificEndpointIdentifier().getLocalEndpointName());

  ConnectionIdentifier connectionIdentifier = event.getConnectionIdentifier();

  this.setConnectionIdentifier(connectionIdentifier.toString());
  String sdp = event.getLocalConnectionDescriptor().toString();

  ContentTypeHeader contentType = null;
  try {
   contentType = headerFactory.createContentTypeHeader("application", "sdp");
  } catch (ParseException ex) {
  }

  String localAddress = provider.getListeningPoints()[0].getIPAddress();
  int localPort = provider.getListeningPoints()[0].getPort();

  Address contactAddress = null;
  try {
   contactAddress = addressFactory.createAddress("sip:" + localAddress + ":" + localPort);
  } catch (ParseException ex) {
  }
  ContactHeader contact = headerFactory.createContactHeader(contactAddress);

  sendRQNT(WELCOME, true);

  Response response = null;
  try {
        response = messageFactory.createResponse(Response.OK, request, contentType,
sdp.getBytes());
  } catch (ParseException ex) {
```

```
   logger.severe("ParseException while trying to create OK Response", ex);
  }

  response.setHeader(contact);
  try {
   txn.sendResponse(response);
  } catch (InvalidArgumentException ex) {
   logger.severe("InvalidArgumentException while trying to send OK Response", ex);
  } catch (SipException ex) {
   logger.severe("SipException while trying to send OK Response", ex);
  }
  break;
 default:
  try {
       response = messageFactory.createResponse(Response.SERVER_INTERNAL_ERROR,
request);
   txn.sendResponse(response);
  } catch (Exception ex) {
   logger.severe("Exception while trying to send SERVER_INTERNAL_ERROR Response", ex);
  }
 }
}

private void sendRQNT(String mediaPath, boolean createActivity) {
       EndpointIdentifier endpointID = new EndpointIdentifier(this.getEndpointName(),
JBOSS_BIND_ADDRESS + ":"
   + MGCP_PEER_PORT);

     NotificationRequest notificationRequest = new NotificationRequest(this, endpointID,
mgcpProvider
   .getUniqueRequestIdentifier());

                        ConnectionIdentifier connectionIdentifier = new
ConnectionIdentifier(this.getConnectionIdentifier());

      EventName[] signalRequests = { new EventName(PackageName.Announcement,
MgcpEvent.ann.withParm(mediaPath), connectionIdentifier) };
 notificationRequest.setSignalRequests(signalRequests);

 RequestedAction[] actions = new RequestedAction[] { RequestedAction.NotifyImmediately };

 RequestedEvent[] requestedEvents = {
       new RequestedEvent(new EventName(PackageName.Announcement, MgcpEvent.oc,
connectionIdentifier), actions),
```

```
        new RequestedEvent(new EventName(PackageName.Announcement, MgcpEvent.of,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf0,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf1,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf2,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf3,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf4,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf5,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf6,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf7,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf8,
connectionIdentifier), actions),

            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmf9,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfA,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfB,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfC,
connectionIdentifier), actions),
            new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfD,
connectionIdentifier), actions),
         new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfStar,
connectionIdentifier), actions),
        new RequestedEvent(new EventName(PackageName.Dtmf, MgcpEvent.dtmfHash,
connectionIdentifier), actions) };

 notificationRequest.setRequestedEvents(requestedEvents);
 notificationRequest.setTransactionHandle(mgcpProvider.getUniqueTransactionHandler());

        NotifiedEntity notifiedEntity = new NotifiedEntity(JBOSS_BIND_ADDRESS,
JBOSS_BIND_ADDRESS, MGCP_PORT);
 notificationRequest.setNotifiedEntity(notifiedEntity);

 if (createActivity) {
```

```
  MgcpEndpointActivity endpointActivity = null;
  try {
   endpointActivity = mgcpProvider.getEndpointActivity(endpointID);
   ActivityContextInterface epnAci = mgcpAcif.getActivityContextInterface(endpointActivity);
   epnAci.attach(sbbContext.getSbbLocalObject());
  } catch (FactoryException ex) {
   ex.printStackTrace();
  } catch (NullPointerException ex) {
   ex.printStackTrace();
  } catch (UnrecognizedActivityException ex) {
   ex.printStackTrace();
  }
 } // if (createActivity)

 mgcpProvider.sendMgcpEvents(new JainMgcpEvent[] { notificationRequest });

 logger.info(" NotificationRequest sent");
}

    public void onNotificationRequestResponse(NotificationRequestResponse event,
ActivityContextInterface aci) {
 logger.info("onNotificationRequestResponse");

 ReturnCode status = event.getReturnCode();

 switch (status.getValue()) {
 case ReturnCode.TRANSACTION_EXECUTED_NORMALLY:
  logger.info("The Announcement should have been started");
  break;
 default:
  ReturnCode rc = event.getReturnCode();
  logger.severe("RQNT failed. Value = " + rc.getValue() + " Comment = " + rc.getComment());

  break;
 }

}

public void onNotifyRequest(Notify event, ActivityContextInterface aci) {
 logger.info("onNotifyRequest");

                    NotificationRequestResponse response = new
NotificationRequestResponse(event.getSource(),
   ReturnCode.Transaction_Executed_Normally);
```

```
response.setTransactionHandle(event.getTransactionHandle());

mgcpProvider.sendMgcpEvents(new JainMgcpEvent[] { response });

EventName[] observedEvents = event.getObservedEvents();

for (EventName observedEvent : observedEvents) {
 switch (observedEvent.getEventIdentifier().intValue()) {
 case MgcpEvent.REPORT_ON_COMPLETION:
  logger.info("Announcemnet Completed NTFY received");
  break;
 case MgcpEvent.REPORT_FAILURE:
  logger.info("Announcemnet Failed received");

  break;
 case MgcpEvent.DTMF_0:
  logger.info("You have pressed 0");
  sendRQNT(DTMF_0, false);
  break;
 case MgcpEvent.DTMF_1:
  logger.info("You have pressed 1");
  sendRQNT(DTMF_1, false);
  break;
 case MgcpEvent.DTMF_2:
  logger.info("You have pressed 2");
  sendRQNT(DTMF_2, false);
  break;
 case MgcpEvent.DTMF_3:
  logger.info("You have pressed 3");
  sendRQNT(DTMF_3, false);
  break;
 case MgcpEvent.DTMF_4:
  logger.info("You have pressed 4");
  sendRQNT(DTMF_4, false);
  break;
 case MgcpEvent.DTMF_5:
  logger.info("You have pressed 5");
  sendRQNT(DTMF_5, false);
  break;
 case MgcpEvent.DTMF_6:
  logger.info("You have pressed 6");
  sendRQNT(DTMF_6, false);
  break;
 case MgcpEvent.DTMF_7:
```

```
    logger.info("You have pressed 7");
    sendRQNT(DTMF_7, false);
    break;
   case MgcpEvent.DTMF_8:
    logger.info("You have pressed 8");
    sendRQNT(DTMF_8, false);
    break;
   case MgcpEvent.DTMF_9:
    logger.info("You have pressed 9");
    sendRQNT(DTMF_9, false);
    break;
   case MgcpEvent.DTMF_A:
    logger.info("You have pressed A");
    sendRQNT(A, false);
    break;
   case MgcpEvent.DTMF_B:
    logger.info("You have pressed B");
    sendRQNT(B, false);
    break;
   case MgcpEvent.DTMF_C:
    logger.info("You have pressed C");
    sendRQNT(C, false);
    break;
   case MgcpEvent.DTMF_D:
    logger.info("You have pressed D");
    sendRQNT(D, false);

    break;
   case MgcpEvent.DTMF_STAR:
    logger.info("You have pressed *");
    sendRQNT(STAR, false);

    break;
   case MgcpEvent.DTMF_HASH:
    logger.info("You have pressed C");
    sendRQNT(POUND, false);

    break;
  }
 }
}

 public void onCallTerminated(RequestEvent evt, ActivityContextInterface aci) {
```

```
        EndpointIdentifier endpointID = new EndpointIdentifier(this.getEndpointName(),
JBOSS_BIND_ADDRESS + ":"
   + MGCP_PEER_PORT);
 DeleteConnection deleteConnection = new DeleteConnection(this, endpointID);

 deleteConnection.setTransactionHandle(mgcpProvider.getUniqueTransactionHandler());
 mgcpProvider.sendMgcpEvents(new JainMgcpEvent[] { deleteConnection });

 ServerTransaction tx = evt.getServerTransaction();
 Request request = evt.getRequest();

 try {
  Response response = messageFactory.createResponse(Response.OK, request);
  tx.sendResponse(response);
 } catch (Exception e) {
  logger.severe("Error while sending DLCX ", e);
 }
}


}
```

Resource Adaptor Implementation

The RA implementation uses the Mobicents JAIN MGCP Implementation.

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
jain.mgcp.PORT	the port to which the MGCP stack should listen	java.lang.Integer	2727
jain.mgcp.IP_ADDRESS	the IP address to which the MGCP stack should attach - if value is not specified the RA will use the underlying Java EE container's bind address address	java.lang.String	

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 MGCPRA. The MGCPRA entity uses the default Resource Adaptor configuration, specified in *Section 3.1, "Configuration"*. The MGCPRA entity is also bound to Resource Adaptor Link Name MGCPRA, to use it in an Sbb add the following XML to its descriptor:.

```
<resource-adaptor-type-binding>
  <resource-adaptor-type-ref>
```

```
            <resource-adaptor-type-name>
               jain-mgcp
            </resource-adaptor-type-name>
            <resource-adaptor-type-vendor>
               net.java
            </resource-adaptor-type-vendor>
            <resource-adaptor-type-version>
               2.0
            </resource-adaptor-type-version>
         </resource-adaptor-type-ref>
         <activity-context-interface-factory-name>
            slee/resources/jainmgcp/2.0/acifactory/demo
         </activity-context-interface-factory-name>
         <resource-adaptor-entity-binding>
            <resource-adaptor-object-name>
               slee/resources/jainmgcp/2.0/provider/demo
            </resource-adaptor-object-name>
            <resource-adaptor-entity-link>
               MGCPRA
            </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 MgcpResourceAdaptor.

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.

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

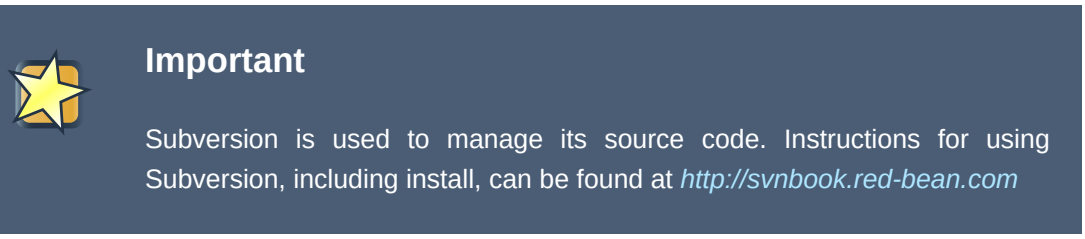
4.1.2. Software Prerequisites

The RA requires JBoss Communications JAIN SLEE properly set.

4.2. JBoss Communications JAIN SLEE MGCP Resource Adaptor Source Code

4.2.1. Release Source Code Building

1. **Downloading the source code**



Use SVN to checkout a specific release source, the base URL is ?, then add the specific release version, lets consider 2.1.0.GA.

```
[usr]$ svn co ?/2.1.0.GA slee-ra-mgcp-2.1.0.GA
```

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-mgcp-2.1.0.GA
[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 MGCP 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 MGCP 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

Clustering is not yet supported by MGCP RA

Appendix A. Revision History

Revision History
Revision 1.0 Mon Feb 08 2010 AmitBhayani
Creation of the JBoss Communications JAIN SLEE MGCP RA User Guide.

Index

F

feedback, viii