JBoss Communications JAIN SLEE MAP Resource Adaptor User Guide

by Amit Bhayani

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 MAP Resource Adaptor**, or contact the authors.

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

Mobile application part (MAP) is the protocol that is used to allow the GSM network nodes within the Network Switching Subsystem to communicate with each other to provide services, such as roaming capability, text messaging (SMS), Unstructured Supplementary Service Data (USSD) and subscriber authentication. MAP provides an application layer on which to build the services that support a GSM network. This application layer provides a standardized set of operations. MAP is transported and encapsulated with the SS7 protocols MTP, SCCP, and TCAP.

For further details please look at specs *http://www.3gpp.org/ftp/Specs/html-info/29002.htm*

This resource adaptor provides a MAP API for JAIN SLEE applications, adapting the MAP specification for USSD.

Resource Adaptor Type

MAP Resource Adaptor Type is defined by JBoss Communications team as part of effort to standardize RA Types.

2.1. Activities

An MAP activity object represents a set of related events in an MAP resource. This RA Type defines only one activity object:

MAPDialog
: All the events related to MAP Dialog and events related to Service are fired on this activity. This activity ends implicitly when MAP stack sends P-Abort or explicitly when user aborts the Dialog or end's the Dialog. Class name is `org.mobicents.protocols.ss7.map.api.MAPDialog`

  New `MAPDialog` activity objects are created via specific MAP Service interface. Check *Section 2.4, "Resource Adaptor Interface"* section for available services.

  Depending on service used, activity object provides additional set of methods. For instance `USSD` dialog: `org.mobicents.protocols.ss7.map.api.service.supplementary.MAPDialogSupplementary` exposes methods specific for exchange of `USSD` messages.

2.2. Events

Events represent's MAP's common services as well as services related to USSD Events are fired on `MAPDialog`

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
: org.mobicents.protocols.ss7.map.

Event Class
: org.mobicents.slee.resource.map.

Version for all defined events is 1.0

Vendor for all defined events is org.mobicents

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

Table 2.1. Dialog events

Name	Event Class	Comments
DIALOG_DELIMITER	DialogDelimiter	Indicates end of MAP commands that triggered other events to be fired.
DIALOG_REQUEST	DialogRequest	Generic event representing ANY map request. This event is fired for ALL incoming requests.
DIALOG_ACCEPT	DialogAccept	Indicates that remote peer acknowledged dialog. This event is fired prior to any other event in such case.
DIALOG_REJECT	DialogReject	Opposite to DIALOG_ACCEPT. Indicates that remote peer rejected dialog for some reason. This event is fired prior to one indicating reason.
DIALOG_USERABORT	DialogUserAbort	Fired when remote MAP user aborts dialog.
DIALOG_PROVIDERABORT	DialogProviderAbort	Fired when when dialog is aborted due to transport level error.
DIALOG_CLOSE	DialogClose	Fired when dialog is closed via TCAP-END primitive.
DIALOG_NOTICE	DialogNotice	Fired when abnormal message is received within dialog. For instance when when duplicated `InvokeID` or wrong operation is received(for running MAP service).
DIALOG_TIMEOUT	DialogTimeout	Fired when dialog is about to timeout. Depending on configuration RA may sustain dialog or let it timeout. This event is fired when there is no activity on dialog for extended period of time.

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
org.mobicents.protocols.ss7.map.

Event Class
org.mobicents.protocols.ss7.map.api.service.supplementary.

Version for all defined events is 1.0

Vendor for all defined events is org.mobicents

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

Table 2.2. Dialog service events

Name	Event Class	Comments
PROCESS_UNSTRUCTURED_SS_REQUEST_INDICATION	ProcessUnstructuredSSIndication	Fired when initial USSD request is received.
UNSTRUCTURED_SS_REQUEST_INDICATION	UnstructuredSSIndication	Fired for each subsequent USSD request.

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
org.mobicents.protocols.ss7.map.

Event Class
org.mobicents.slee.resource.map.

Version for all defined events is 1.0

Vendor for all defined events is org.mobicents

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

Table 2.3. Component events

Name	Event Class	Comments
INVOKE_TIMEOUT	InvokeTimeout	Fired when localy initiated Invoke does not receive any answer for extended period of time.
ERROR_COMPONENT	ErrorComponent	Fired when remote peer indicates abnormal component. It indicates some protocol error in component sent from local peer.
PROVIDER_ERROR_COMPONENT	ProviderErrorComponent	Fired when transport provider encounters error on parsing MAP message. It is similar in meaning to ERROR_COMPONENT, but for component received from remote peer.
REJECT_COMPONENT	RejectComponent	Fired when remote end rejects component for some reason.

2.3. Activity Context Interface Factory

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

```
package org.mobicents.slee.resource.map;

import org.mobicents.protocols.ss7.map.api.MAPDialog;

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

public interface MAPContextInterfaceFactory {

    public ActivityContextInterface getActivityContextInterface(MAPDialog dialog) throws
 NullPointerException,
   UnrecognizedActivityException, FactoryException;

}
```

2.4. Resource Adaptor Interface

The MAP Resource Adaptor SBB Interface provides SBBs with access to the MAP objects required for creating a new, aborting, ending a MAPdialog and sending USSD Request/Response. It is defined as follows:

```
package org.mobicents.protocols.ss7.map.api;


public interface MAPProvider {

  public abstract void addMAPDialogListener(MAPDialogListener mapdialoglistener);

    public abstract void removeMAPDialogListener(MAPDialogListener mapdialoglistener);

    public abstract MapServiceFactory getMapServiceFactory();

    public abstract MAPErrorMessageFactory getMAPErrorMessageFactory();

    public abstract MAPDialog getMAPDialog(Long long1);

    public abstract MAPServiceSupplementary getMAPServiceSupplementary();

    public abstract MAPServiceSms getMAPServiceSms();

    public abstract MAPServiceLsm getMAPServiceLsm();

}
```

public abstract void addMAPDialogListener(MAPDialogListener mapdialoglistener);
this method is not supported. Call to it causes NotSupportedException to be thrown.

public abstract void removeMAPDialogListener(MAPDialogListener mapdialoglistener);
this method is not supported. Call to it causes NotSupportedException to be thrown.

public abstract MapServiceFactory getMapServiceFactory();
retrieves factory for generic MAP components(interface name is missleading, will be changed in CR1)

public abstract MAPErrorMessageFactory getMAPErrorMessageFactory();
retrieves implementation of MAP error message factory. Error messages are used to indicate erroneous conditions.

public abstract MAPDialog getMAPDialog(Long dialogId);
retrieves active dialog by its `ID`.

public abstract MAPServiceSupplementary getMAPServiceSupplementary();
retrieves MAP sumplementary service. It is used to create USSD dialogs.

public abstract MAPServiceSms getMAPServiceSms();
retrieves MAP SMS service. It is used to create SMS dialogs. In current release it is not supported.

public abstract MAPServiceLsm getMAPServiceLsm();
retrieves MAP LMS service. It is used to create LMS dialogs. In current release it is not supported.

Note

As MAP stack is beeing completed, it will support more services, this list of `getMAPServiceX` will expand to support all implemented services.

2.5. Restrictions

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

2.6. Sbb Code Examples

The following code shows complete flow of application receiving the MAP Dialog request and then USSD Request. Application sends back Unstructured SS Response and finally on receiving Unstructured SS Request, application closes the MAPDialog

```
public abstract class SipSbb implements Sbb {

 private SbbContext sbbContext;


 private MAPContextInterfaceFactory mapAcif;
 private MAPProvider mapProvider;
 private MapServiceFactory mapServiceFactory;
```

```
private static byte ussdDataCodingScheme = 0x0F;


private Tracer logger;

/** Creates a new instance of CallSbb */
public SipSbb() {
}

/**
 * MAP USSD Event Handlers
 */

public void onProcessUnstructuredSSRequest(
  ProcessUnstructuredSSIndication evt, ActivityContextInterface aci) {

 try {

  long invokeId = evt.getInvokeId();
  this.setInvokeId(invokeId);

  String ussdString = evt.getUSSDString().getString();
  this.setUssdString(ussdString);

  int codingScheme = evt.getUSSDDataCodingScheme() & 0xFF;
  String msisdn = evt.getMSISDNAddressString().getAddress();

  if (this.logger.isFineEnabled()) {
   this.logger
     .fine("Received PROCESS_UNSTRUCTURED_
     SS_REQUEST_INDICATION for MAP Dialog Id "
       + evt.getMAPDialog().getDialogId()+
       " ussdString = "+ussdString);
  }

  USSDString ussdStringObj = this.mapServiceFactory
   .createUSSDString("1. Movies 2. Songs 3. End");

  evt.getMAPDialog().addUnstructuredSSResponse(invokeId, false,
    ussdDataCodingScheme, ussdStringObj);

  evt.getMAPDialog().send();
```

```
 } catch (Exception e) {
  logger.severe("Error while sending MAP USSD message", e);
 }

}

public void onUnstructuredSSRequest(UnstructuredSSIndication evt,
  ActivityContextInterface aci) {

 if (this.logger.isFineEnabled()) {
  this.logger
    .fine("Received UNSTRUCTURED_SS_REQUEST_INDICATION for MAP Dialog Id "
      + evt.getMAPDialog().getDialogId());
 }

 try{

 MAPDialog mapDialog = evt.getMAPDialog();
 USSDString ussdStrObj = evt.getUSSDString();

 long invokeId = evt.getInvokeId();


 USSDString ussdStringObj = this.mapServiceFactory.createUSSDString("Thank you");

 evt.getMAPDialog().addUnstructuredSSResponse(invokeId, false,
   ussdDataCodingScheme, ussdStringObj);

 //End MAPDialog
 evt.getMAPDialog().close(false);

 }catch(Exception e){
  logger.severe("Error while sending MAP USSD ", e);
 }
}

...


public void setSbbContext(SbbContext sbbContext) {
 this.sbbContext = sbbContext;
 this.logger = sbbContext.getTracer("USSD-SIP");

 try {
```

```
  Context ctx = (Context) new InitialContext()
    .lookup("java:comp/env");



  mapAcif = (MAPContextInterfaceFactory) ctx
    .lookup("slee/resources/map/2.0/acifactory");

  mapProvider = (MAPProvider) ctx
    .lookup("slee/resources/map/2.0/provider");

  this.mapServiceFactory = this.mapProvider.getMapServiceFactory();


 } catch (Exception ne) {
  logger.severe("Could not set SBB context:", ne);
 }
}

public void unsetSbbContext() {
 this.sbbContext = null;
 this.logger = null;
}

public void sbbCreate() throws CreateException {
}

public void sbbPostCreate() throws CreateException {
}

public void sbbActivate() {
}

public void sbbPassivate() {
}

public void sbbLoad() {
}

public void sbbStore() {
}

public void sbbRemove() {
}
```

```
 public void sbbExceptionThrown(Exception exception, Object object,
   ActivityContextInterface activityContextInterface) {
 }

 public void sbbRolledBack(RolledBackContext rolledBackContext) {
 }
}
```

Resource Adaptor Implementation

The RA implementation uses the JBoss Communications MAP stack. The stack is the result of the work done by JBoss Communications Media Server 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. It supports following properties:

Table 3.1. Resource Adaptor's Configuration Properties - map-default-ra.properties

Property Name	Description	Property Type	Default Value
ssn	Sub-System Number to be used for transport.	java.lang.Interge	8
sccpJndi	JNDI name of SCCP service to be used.	java.lang.String	java:/mobicents/ss7/sccp
timeout	Controls how many times dialog can timeout. '-1' - indicates that dialog will never be released after timeout, othen non negative values indicates how many timeouts may occur before dialog is released.	java.lang.Integer	

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

The `MAPRA` entity is also bound to Resource Adaptor Link Name `MAPRA`, 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>
         MAPResourceAdaptorType
      </resource-adaptor-type-name>
      <resource-adaptor-type-vendor>
         org.mobicents
      </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/map/2.0/acifactory
   </activity-context-interface-factory-name>
   <resource-adaptor-entity-binding>
      <resource-adaptor-object-name>
         slee/resources/map/2.0/provider
      </resource-adaptor-object-name>
      <resource-adaptor-entity-link>
         MAPRA
      </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 `MAPResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=MAPRA]`

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.

Of course, memory is only needed to store the Resource Adaptor state, the faster the CPU more MAP Messages processing is 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 MAP 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 2.4.2.BETA1.

```
[usr]$ svn co ?/2.4.2.BETA1 slee-ra-MAP-2.4.2.BETA1
```

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-MAP-2.4.2.BETA1
[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 MAP 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 MAP 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=`.

Appendix A. Revision History

Revision History
Revision 1.0 Tue Dec 30 2009 AmitBhayani
Creation of the JBoss Communications JAIN SLEE MAP RA User Guide.

Index

F