User Guide to Restcomm jSS7 CAP RA 7.0.0-SNAPSHOT

Table of Contents

Preface

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 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.

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 to switch to the first virtual terminal. Press 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 menu:>[] shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select from the **Preferences** | **] sub-menu in the menu: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.

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"));
   }

}
```

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.

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 {this-issue.tracker.ur}, against the product Restcomm jSS7` `, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: Restcomm jSS7

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.

Chapter 1. Introduction to Restcomm JAIN SLEE CAP Resource Adaptor

The CAMEL (Customized Applications for Mobile network Enhanced Logic) Application Part (CAP) protocol is used by network operators to provide their subscribers with operator specific services even when roaming outside the HPLMN. CAP provides services such as prepaid roaming services, fraud control, special numbers and closed user groups (e.g., office extension numbers that work everywhere). CAP has been defined in four versions (phases), each of which has an accompanying specification that builds upon the previous phase. This application layer provides a standardized set of services. uses the services of the SS7network, specifically the Signaling Connection Control Part (SCCP) and the Transaction Capabilities Application Part (TCAP)

For better understanding of CA read the specifications defined in 3GPP TS 22.078 (service aspects) and 3GPP TS 23.078 (technical realization).

This resource adaptor provides a CAP API for JAIN SLEE applications, adapting the CAP Stack defined in Restcomm jSS7.

Chapter 2. Resource Adaptor Type

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

2.1. Activities

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

CAPDialog

All the events related to CAP Dialog and events related to Service are fired on this activity. This activity ends implicitly when CAP stack sends P-Abort or explicitly when user aborts the Dialog or end's the Dialog. Class name is `org.mobicents.protocols.ss7.cap.api.CAPDialog` New CAPDialog activity objects are created via specific CAP Service interface. Check Resource Adaptor Interface section for available services. Depending on service used, activity object provides additional set of methods. For instance Circuit Switched Call dialog: org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPDialogCircuitSwitchedCall exposes methods specific for exchange of call related messages.

2.2. Events

Events represent's CAP's messages related to dialog, component as well as services. Events are fired on `CAPDialog`. Below sections detail different type of events, depending on cause of it being fired into SLEE.

2.2.1. Component

Below events are fired into when something happens with components passed in messages.

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

Name

ss7.cap.

Event Class

org.mobicents.slee.resource.cap.events.

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 1. Component events

Name	Event Class	Comments
INVOKE_TIMEOUT	InvokeTimeout	Fired when locally 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.
REJECT_COMPONENT	RejectComponent	Fired when remote end rejects component for some reason.

2.2.2. Dialog

Dialog events are fired into SLEE to indicate basic occurrence of dialog related data.

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

Name

ss7.cap.

Event Class

org.mobicents.slee.resource.cap.events

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. Dialog events

Name	Event Class	Comments
DIALOG_DELIMITER	DialogDelimiter	Indicates end of CAP commands that triggered other events to be fired.
DIALOG_REQUEST	DialogRequest	Generic event representing ANY cap 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.

Name	Event Class	Comments
DIALOG_USERABORT	DialogUserAbort	Fired when remote CAP 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.
DIALOG_RELEASE	DialogRelease	Fired when Dialog and all the resources related to dialog are released. This is last event on this activity after which activity will end.

2.2.3. Circuit Switched Call

Below events are fired when dialog receives callbacks for circuit switch call service.

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

Name

ss7.cap.service.circuitSwitchedCall.

Event Class

org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.

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 3. Mobility - Location Management Service events

Name	Event Class	Comments
INITIAL_DP_REQUEST	InitialDPRequest	The SSF will send InitialDPRequest to SCF to start CAMEL service.
ACTIVITY_TEST_REQUEST	ActivityTestRequest	The activity test (AT) is used for testing the CAMEL dialogue between the gsmSCF and the gsmSSF. The SCP may send CAP AT at regular intervals to the gsmSSF, e.g. every 15 min. The only function of CAP AT is to verify the existence of the CAMEL dialogue. When the gsmSSF receives CAP AT, it returns an empty RESULT to the gsmSCF. If the gsmSCF does not receive an operation RESULT within the operation time for CAP AT, e.g. 5 s, then the gsmSCF terminates the CAMEL service. CAP AT is normally sent by the SCP platform, not by the CAMEL service. The arrival of CAP AT in the gsmSSF has no impact on any call handling process or on the BCSM. The sending of CAP AT is not dependent on the phase of the call or on the gsmSSF FSM state
ACTIVITY_TEST_RESPONSE	ActivityTestResponse	Response to ActivityTestRequest
APPLY_CHARGING_ REPORT_REQUEST	ApplyChargingReportRequest	This is the report that is sent from gsmSSF to gsmSCF at the end of a call period or when the call is released. In addition, when call set up failure occurs, such as called party busy or no answer, the gsmSSF also sends a charging report (if previously requested).
APPLY_CHARGING_REQUEST	ApplyChargingRequest	This is the instruction from the gsmSCF to the gsmSSF to start or continue monitoring the call duration.
ASSIST_REQUEST_ INSTRUCTIONS_REQUEST	AssistRequest InstructionsRequest	Used by assisting gsmSSF to start the assisting dialogue.

Name	Event Class	Comments
CALL_INFORMATION_ REPORT_REQUEST	CallInformationReportRequest	This event contains the call-related information that was previously requested for this call.
CALL_INFORMATION_ REQUEST_REQUEST	CallInformationRequestRequest	This operation is used to request call-related information.
CANCEL_REQUEST	CancelRequest	This event has a dual purpose: * It may be used to disarm armed detection points and to cancel requests for reports. It is normally used when a CAMEL service wants to terminate the relationship * It may be used to prevent or stop the execution of a user interaction operation, which was previously sent to the gsmSRF or to the intelligent peripheral.
CONNECT_REQUEST	ConnectRequest	This event is used to instruct the gsmSSF to continue call establishment with modified information. This operation may also be used to generate a follow-on call.
CONNECT_TO_ RESOURCE_REQUEST	ConnectToResourceRequest	This event is used to instruct the gsmSSF or assisting gsmSSF to connect the call to a specialized resource, for user interaction
CONTINUE_REQUEST	ContinueRequest	This operation is used to instruct the gsmSSF to continue call processing at the DP where call processing was suspended
DISCONNECT_FORWARD_ CONNECTION_REQUEST	DisconnectForward ConnectionRequest	This operation is used to terminate the connection to a specialized resource or to terminate a temporary connection to an assisting gsmSSF or intelligent peripheral
ESTABLISH_TEMPORARY_ CONNECTION_REQUEST	EstablishTemporary ConnectionRequest	This event is used to establish a temporary connection between the serving (G)MSC and an MSC with assisting gsmSSF

Name	Event Class	Comments
EVENT_REPORT_BCSM_REQUEST	EventReportBCSMRequest	This event is used by the gsmSSF to inform the gsmSCF about the occurrence of an event
FURNISH_CHARGING_INFORMATION_REQUEST	FurnishChargingInformationRequest	The gsmSCF may use this operation to place service-specific data in the CDR for the call
PLAY_ANNOUNCEMENT_REQUEST	PlayAnnouncementRequest	A CAMEL service may use this operation to instruct the gsmSRF or intelligent peripheral to play an announcement
RELEASE_CALL_REQUEST	ReleaseCallRequest	This operation is used by the gsmSCF to release a call
PROMPT_AND_COLLECT_USER_INFORMATION_REQUEST	PromptAndCollectUserInformationRequest	A CAMEL service may use this operation to instruct the gsmSRF or intelligent peripheral to play an announcement and to collect digits from the user
PROMPT_AND_COLLECT_USER_INFORMATION_RESPONSE	PromptAndCollectUserInformationResponse	Response to PromptAndCollectUserInformationRequest
REQUEST_REPORT_BCSM_EVENT_REQUEST	RequestReportBCSMEventRequest	This operation may be used by the gsmSCF to arm or disarm detection points in the BCSM
RESET_TIMER_REQUEST	ResetTimerRequest	This operation may be used by the gsmSCF to reload and restart the Tssf timer
SEND_CHARGING_INFORMATION_REQUEST	SendChargingInformationRequest	This operation may be used by the gsmSCF to send advice of charge information to the served subscriber
SPECIALIZED_RESOURCE_REPORT_REQUEST	SpecializedResourceReportRequest	This operation is used by the gsmSRF to inform the gsmSCF that the playing of an announcement is complete

2.3. Activity Context Interface Factory

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

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

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

import org.mobicents.protocols.ss7.cap.api.CAPDialog;

public interface CAPContextInterfaceFactory {

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

}
```

2.4. Resource Adaptor Interface

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

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

import java.io.Serializable;

import org.mobicents.protocols.ss7.cap.api.errors.CAPErrorMessageFactory;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPServiceCircuitSwitc
hedCall;
import org.mobicents.protocols.ss7.cap.api.service.gprs.CAPServiceGprs;
import org.mobicents.protocols.ss7.cap.api.service.sms.CAPServiceSms;
import org.mobicents.protocols.ss7.inap.api.INAPParameterFactory;
import org.mobicents.protocols.ss7.isup.ISUPParameterFactory;
import org.mobicents.protocols.ss7.map.api.MAPParameterFactory;

public interface CAPProvider extends Serializable {

    /**
     * Add CAP Dialog listener to the Stack
     *
     * @param capDialogListener
     */
    public void addCAPDialogListener(CAPDialogListener capDialogListener);

    /**
     * Remove CAP DIalog Listener from the stack
     *
     * @param capDialogListener
```

```
     */
    public void removeCAPDialogListener(CAPDialogListener capDialogListener);

    /**
     * Get the {@link CAPParameterFactory}
     *
     * @return
     */
    public CAPParameterFactory getCAPParameterFactory();

    /**
     * Get the {@link MAPParameterFactory}
     *
     * @return
     */
    public MAPParameterFactory getMAPParameterFactory();

    /**
     * Get the {@link ISUPParameterFactory}
     *
     * @return
     */
    public ISUPParameterFactory getISUPParameterFactory();

    /**
     * Get the {@link INAPParameterFactory}
     *
     * @return
     */
    public INAPParameterFactory getINAPParameterFactory();

    /**
     * Get the {@link CAPErrorMessageFactory}
     *
     * @return
     */
    public CAPErrorMessageFactory getCAPErrorMessageFactory();

    /**
     * Get {@link CAPDialog} corresponding to passed dialogId
     *
     * @param dialogId
     * @return
     */
    public CAPDialog getCAPDialog(Long dialogId);

    public CAPServiceCircuitSwitchedCall getCAPServiceCircuitSwitchedCall();
    public CAPServiceGprs getCAPServiceGprs();
    public CAPServiceSms getCAPServiceSms();

}
```

public void addCAPDialogListener(CAPDialogListener capDialogListener);

this method is not supported. Call to it causes NotSupportedException to be thrown.

public void removeCAPDialogListener(CAPDialogListener capDialogListener);

this method is not supported. Call to it causes NotSupportedException to be thrown.

public CAPParameterFactory getCAPParameterFactory();

retrieves factory for generic CAP components

public abstract MAPParameterFactory getMAPParameterFactory();

retrieves factory for generic MAP components

public ISUPParameterFactory getISUPParameterFactory();

retrieves factory for generic ISUP components

public INAPParameterFactory getINAPParameterFactory();

retrieves factory for generic INAP components

public CAPErrorMessageFactory getCAPErrorMessageFactory();

retrieves implementation of CAP error message factory. Error messages are used to indicate erroneous conditions.

public CAPDialog getCAPDialog(Long dialogId);

retrieves active dialog by its `ID`.

public CAPServiceCircuitSwitchedCall getCAPServiceCircuitSwitchedCall();

retrieves CAP Circuit Switch Call service. It is used to create circuit switch call related dialogs.

public CAPServiceGprs getCAPServiceGprs();

retrieves CAP GPRS Service. It is used to create GPPRS handling dialogs.

```
NOTE: This service is not yet implemented
```

public CAPServiceSms getCAPServiceSms();

retrieves CAP SMS service. It is used to create SMS dialogs.

```
NOTE: This service is not yet implemented
```

As CAP stack is being completed, it will support more services, this list of `getCAPServiceX` will expand to support all implemented services.

2.5. Restrictions

The resource adaptor implementation should prevent SBBs from adding themselves as CAP listeners, or changing the CAP 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 CAP Dialog request and then IDP Request.

```
public abstract class IdpSbb extends MissCallAlertBaseSbb implements Idp {

    public void onDIALOG_DELIMITER(org.mobicents.slee.resource.cap.events
.DialogDelimiter event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_REQUEST(org.mobicents.slee.resource.cap.events.DialogRequest
event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_ACCEPT(org.mobicents.slee.resource.cap.events.DialogAccept
event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_USERABORT(org.mobicents.slee.resource.cap.events
.DialogUserAbort event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_PROVIDERABORT(org.mobicents.slee.resource.cap.events
.DialogProviderAbort event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_CLOSE(org.mobicents.slee.resource.cap.events.DialogClose
event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_NOTICE(org.mobicents.slee.resource.cap.events.DialogNotice
event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_TIMEOUT(org.mobicents.slee.resource.cap.events.DialogTimeout
event,
            ActivityContextInterface aci) {
    }

    public void onDIALOG_RELEASE(org.mobicents.slee.resource.cap.events.DialogRelease
```

```
event,
            ActivityContextInterface aci) {
    }

    public void onINITIAL_DP_REQUEST(
            org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall
.InitialDPRequest event,
            ActivityContextInterface aci) {

        if (this.logger.isFineEnabled()) {
            this.logger.fine("INITIAL_DP_REQUEST = " + event);
        }

        try {
            // TODO All processing
            CalledPartyNumberCap calledPartyNumberCap = event.getCalledPartyNumber();
            CallingPartyNumberCap callingPartyNumberCap = event.
getCallingPartyNumber();

            CalledPartyBCDNumber calledPartyBCDNumber = event.
getCalledPartyBCDNumber();
            AddressNature addressNature = calledPartyBCDNumber.getAddressNature();
            NumberingPlan numberingPlan = calledPartyBCDNumber.getNumberingPlan();
            String destinationMSISDN = calledPartyBCDNumber.getAddress();

            .......
            .......

        } catch (Exception e) {
            logger.severe("Unexpected error: ", e);
            // TODO: isolate try+catch per if/else
            // TODO: terminater dialog
        }
    }

    public void onINVOKE_TIMEOUT(org.mobicents.slee.resource.cap.events.InvokeTimeout
event,
            ActivityContextInterface aci) {
    }

    public void onERROR_COMPONENT(org.mobicents.slee.resource.cap.events
.ErrorComponent event,
            ActivityContextInterface aci) {
    }

    public void onREJECT_COMPONENT(org.mobicents.slee.resource.cap.events
.RejectComponent event,
            ActivityContextInterface aci) {
    }

    public void setSbbContext(SbbContext context) {
```

```
        super.setSbbContext(sbbContext);

        this.logger = sbbContext.getTracer(getClass().getName());

        // CAP RA
        this.capAcif = (CAPContextInterfaceFactory) this.sbbContext
            .getActivityContextInterfaceFactory(capRATypeID);
        this.capProvider = (CAPProvider) this.sbbContext
            .getResourceAdaptorInterface(capRATypeID, capRaLink);
        this.capParameterFactory = this.capProvider.getCAPParameterFactory();
    }

}
```

Chapter 3. Resource Adaptor Implementation

The RA implementation uses the Restcomm CAP stack. The stack is the result of the work done by Restcomm JSLEE 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 4. Resource Adaptor's Configuration Properties - cap-default-ra.properties

Property Name	Description	Property Type	Default Value
mapJndi	JNDI name of CAP stack	java.lang.String	java:/mobicents/ss7/cap

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 `CAPRA`. The `CAPRA` entity uses the default Resource Adaptor configuration, specified in Configuration.

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

3.3.2. Alarms

No alarms are set by this Resource Adaptor.

Chapter 4. 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 CAP Messages processing is supported, yet no particular CPU is a real requirement to use the RA.

4.1.2. Software Prerequisites

The RA requires Restcomm JAIN SLEE properly set.

4.2. Restcomm JAIN SLEE CAP Resource Adaptor Source Code

4.2.1. Release Source Code Building

1. Downloading the source code

Git is used to manage Restcomm JAIN SLEE source code. Instructions for downloading, installing and using Git can be found at http://git-scm.com/

Use Git to checkout a specific release source, the Git repository URL is https://github.com/Restcomm/jain-slee.ss7, then switch to the specific release version, lets consider 7.0.0-SNAPSHOT.

```
[usr]$ git clone https://github.com/Restcomm/jain-slee.ss7/
[usr]$ cd jain-slee.ss7
[usr]$ git checkout tags/{project-version}
```

2. Building the source code

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 resources/CAP
[usr]$ mvn install
```

Once the process finishes you should have the `deployable-unit` jar file in the *target* directory, if Restcomm JAIN SLEE is installed and environment variable JBOSS_HOME is pointing to its underlying {jee.platform} directory, then the deployable unit jar will also be deployed in the container.

4.2.2. Development Master Source Building

Similar process as for Release Source Code Building, the only change is the Git reference should be the `master`. The `git checkout tags/7.0.0-SNAPSHOT` command should not be performed. If already performed, the following should be used in order to switch back to the master:

```
[usr]$ git checkout master
```

4.3. Installing Restcomm JAIN SLEE CAP 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` Restcomm JAIN SLEE server profile deploy directory, to deploy to another server profile use the argument `-Dnode=`.

4.4. Uninstalling Restcomm JAIN SLEE CAP 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` Restcomm JAIN SLEE server profile deploy directory, to undeploy from another server profile use the argument `-Dnode=`.

Appendix A: Revision History