User Guide to Restcomm jSS7 TCAP RA 2.8.0-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 TCAP Resource Adaptor

The Transaction Capabilities Application Part (TCAP) is defined in ITU-T Recommendations Q.771-Q.775. TCAP allows services at network nodes to communicate with each other using an agreed-upon set of data elements. The primary purpose of TCAP is to facilitate multiple concurrent dialogs between the same sub-systems on the same machines, using Transaction IDs to differentiate these, similar to the way TCP ports facilitate multiplexing connections between the same IP addresses on the Internet.

For better understanding of TCAP read the specifications defined in ITU-T Recommendations Q.771-Q.775

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

Chapter 2. Resource Adaptor Type

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

2.1. Activities

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

Dialog

All the events related to TCAP Dialog and events related to Service are fired on this activity. This activity ends implicitly when TCAP stack sends P-Abort or explicitly when user aborts the Dialog or end's the Dialog. Class name is `org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog` New Dialog activity objects is created via SBB interface exposed by Resource Adaptor

2.2. Events

Events represent's TCAP 's messages related to dialog and components. Events are fired on `Dialog`. 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.tcap.

Event Class

org.mobicents.slee.resource.tcap.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
COMPONENT_INVOKE	InvokeEvent	Fired when Invoke is received by underlying TCAP stack

Name	Event Class	Comments
COMPONENT_INVOKE_ TIMEOUT	InvokeEvent	Fired when locally initiated Invoke does not receive any answer for extended period of time.
COMPONENT_REJECT	RejectEvent	Fired when remote end rejects component for some reason.
COMPONENT_ RETURNRESULT	ReturnResultEvent	Fired when remote end responded to invoke sent earlier indicating that there are more response to arrive.
COMPONENT_ RETURNRESULT_LAST	ReturnResultLastEvent	Fired when remote end responded to invoke sent earlier indicating that this is last response.
COMPONENT_ RETURNERROR	ReturnErrorEvent	Fired when remote peer indicates abnormal component. It indicates some protocol error in component sent from local peer.

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.tcap.

Event Class

org.mobicents.protocols.ss7.tcap.api.tc.dialog.

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_UNI	Dialog	Indicates TCAP stack received uni-direction dialog. Dialog will be released automatically

Name	Event Class	Comments
DIALOG_BEGIN	Dialog	Event indicate new Dialog is initiated by peer.
DIALOG_CONTINUE	Dialog	Event indicates continuation of existing dialog
DIALOG_END	Dialog	Event indicates peer ended dialog
DIALOG_USERABORT	events.TCUserAbortIndication	Peer user aborted dialog
DIALOG_PROVIDERABORT	events.TCPAbortIndication	Either local stack or peer stack aborted dialog
DIALOG_NOTICE	Dialog	Fired when abnormal message is received within dialog. For instance when when duplicated InvokeID or wrong operation is received.
DIALOG_RELEASED	Dialog	Fired when Dialog and all the resources related to dialog are released. This is last event on this activity after which activity will end.
DIALOG_TIMEOUT	Dialog	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_DELIMITER	Dialog	Indicates all the component events are fired.

2.3. Activity Context Interface Factory

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

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

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

import org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog;

public interface TCAPContextInterfaceFactory {

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

}
```

2.4. Resource Adaptor Interface

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

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

import java.io.Serializable;

import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog;

public interface TCAPProvider extends Serializable {

    /**
     * Create new structured dialog.
     * @param localAddress - desired local address
     * @param remoteAddress - initial remote address, it can change after first
TCContinue.
     * @return
     */
    public Dialog getNewDialog(SccpAddress localAddress, SccpAddress remoteAddress)
throws TCAPException;
    /**
     * Create new unstructured dialog.
     * @param localAddress
     * @param remoteAddress
     * @return
     * @throws TCAPException
     */
    public Dialog getNewUnstructuredDialog(SccpAddress localAddress, SccpAddress
remoteAddress) throws TCAPException;

    ///////////////
    // Factories //
    ///////////////

    public DialogPrimitiveFactory getDialogPrimitiveFactory();
    public ComponentPrimitiveFactory getComponentPrimitiveFactory();

    ///////////////
    // Listeners //
    ///////////////

    public void addTCListener(TCListener lst);

    public void removeTCListener(TCListener lst);

    public boolean getPreviewMode();
}
```

public void addTCListener(TCListener lst);

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

public void removeTCListener(TCListener lst);

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

public DialogPrimitiveFactory getDialogPrimitiveFactory();

retrieves factory for creating Dialog primitives

public ComponentPrimitiveFactory getComponentPrimitiveFactory();

retrieves factory for creating components

public Dialog getNewDialog(SccpAddress localAddress, SccpAddress remoteAddress) throws TCAPException;

Creates new structured Dialog

public Dialog getNewUnstructuredDialog(SccpAddress localAddress, SccpAddress remoteAddress) throws TCAPException;

Creates new unstructured dialog

public boolean getPreviewMode();

Returns true if this TCAP stack is configured as preview

2.5. Restrictions

The resource adaptor implementation should prevent SBBs from adding themselves as TCAP listeners, or changing the TCAP 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.

```
package org.mobicents.example;

import javax.slee.*;
import org.mobicents.slee.*;

public abstract class TCAPExampleSbb implements Sbb, TCAPExample {

    public void onDIALOG_UNI(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_BEGIN(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }
```

```
    public void onDIALOG_CONTINUE(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_END(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_USERABORT(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCUserAbortIndication
event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_PROVIDERABORT(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCPAbortIndication
event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_NOTICE(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_RELEASED(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_TIMEOUT(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onDIALOG_DELIMITER(
        org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onCOMPONENT_INVOKE(
        org.mobicents.slee.resource.tcap.events.InvokeEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onCOMPONENT_INVOKE_TIMEOUT(
        org.mobicents.slee.resource.tcap.events.InvokeEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }
```

```
    public void onCOMPONENT_REJECT(
        org.mobicents.slee.resource.tcap.events.RejectEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onCOMPONENT_RETURNRESULT(
        org.mobicents.slee.resource.tcap.events.ReturnResultEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onCOMPONENT_RETURNRESULT_LAST(
        org.mobicents.slee.resource.tcap.events.ReturnResultLastEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }

    public void onCOMPONENT_RETURNERROR(
        org.mobicents.slee.resource.tcap.events.ReturnErrorEvent event,
        ActivityContextInterface aci/*, EventContext eventContext*/) {
    }



    // TODO: Perform further operations if required in these methods.
    public void setSbbContext(SbbContext context) { this.sbbContext = (SbbContextExt)
context; }
    public void unsetSbbContext() { this.sbbContext = null; }

    // TODO: Implement the lifecycle methods if required
    public void sbbCreate() throws javax.slee.CreateException {}
    public void sbbPostCreate() throws javax.slee.CreateException {}
    public void sbbActivate() {}
    public void sbbPassivate() {}
    public void sbbRemove() {}
    public void sbbLoad() {}
    public void sbbStore() {}
    public void sbbExceptionThrown(
        Exception exception, Object event,
        ActivityContextInterface activity) {}
    public void sbbRolledBack(RolledBackContext context) {}



    /**
     * Convenience method to retrieve the SbbContext object stored in setSbbContext.
     *
     * TODO: If your SBB doesn't require the SbbContext object you may remove this
     * method, the sbbContext variable and the variable assignment in setSbbContext().
     *
     * @return this SBB's SbbContext object
     */
```

```
    protected SbbContextExt getSbbContext() {
        return sbbContext;
    }

    private SbbContextExt sbbContext; // This SBB's SbbContext

}
```

Chapter 3. Resource Adaptor Implementation

The RA implementation uses the Restcomm TCAP 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 3. Resource Adaptor's Configuration Properties - tcap-default-ra.properties

Property Name	Description	Property Type	Default Value
tcapJndi	JNDI name of TCAP stack	java.lang.String	java:/mobicents/ss7/tcap

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

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

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 TCAP 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 TCAP 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 2.8.0-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/TCAP
[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/2.8.0-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 TCAP 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 TCAP 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