User Guide to Restcomm jSS7 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

1.1. Restcomm jSS7

Restcomm jSS7 is the only Open Source Java based implementation of the SS7 protocol stack. It provides implementation for MTP2, MTP3, ISUP, SCCP, TCAP, CAMEL (Phase I, Phase II, Phase III and Phase IV) and MAP protocols for a dedicated equipment. It also has in-built support for SIGTRAN (M3UA) over IP and strictly adheres to the standards and specifications defined by the International Telecommunications Union (ITU) and American National Standards Institute (ANSI). The platform offers developers with a flexible API set that hides the lower layer details (legacy SS7 links or SIGTRAN) and therefore makes it simple and easy to develop SS7 applications as well as to migrate your applications from TDM equipments to M3UA. Restcomm jSS7 is based on an easily scalable and configurable load-balancing architecture.

Restcomm jSS7 supports TDM hardware offered by major vendors in the market, namely Intel family boards (Dialogic) and Zaptel/Dahdi (Digium, Sangoma).

If you intend to use only M3UA you can install the jSS7 on any Operating System that supports Java. However if you wish to use SS7 cards, the native libaries for these are only compiled for Linux at the moment. Restcomm jSS7 can work as a standalone library. However if you wish to use JSLEE Resource Adapters, the Command Line Interface (Shell Management tool) or the GUI Management Console for run-time configuration, then you must have JBoss Application Server installed and running. Restcomm jSS7 comes with JSLEE TCAP, MAP, CAP and ISUP Resource Adaptors (RA) that enable developers to build SS7 applications with ease. Developers only require an understanding of Resource Adaptors and can focus on building applications quickly and efficiently rather than worry about the SS7 stack.

The Open Source Software gives you the flexibility to understand the readily available source code and customise the product for your Enterprise needs.

This guide provides details on configuring and using the platform and information regarding the supported protocols and compliant standards. For installation instructions, please refer to the Installation Guide published along with this.

1.2. About SS7

SS7 (Signaling System No.7) is a set of signaling protocols defined for information exchange in telephony. It is the global standard for telecommunications and is defined by the International Telecommunication Union (ITU) - Telecommunication Standardization Sector (ITU-T). It is also commonly referred to as Common Channel Signaling System No. 7 (i.e., SS7 or C7).

In telephony (wireless and wireline), the information associated with a call must be exchanged between a telephone and the telephone exchange or between exchanges, transit nodes and other elements in the network. Information exchange is required to set up, control and tear down calls and this exchange of information is called Signaling. SS7 is the global standard that defines the procedures and protocol to be followed by network elements in the Public Switched Telephone Network (PSTN) in order to exchange information over a digital signaling network to effect wireless

(cellular) and wireline call setup, route,control, monitor and terminate.

The ITU definition of SS7 allows for national variants such as the American National Standards Institute (ANSI) and Bell Communications Research (Telcordia Technologies) standards used in North America and the European Telecommunications Standards Institute (ETSI) standard used in Europe.

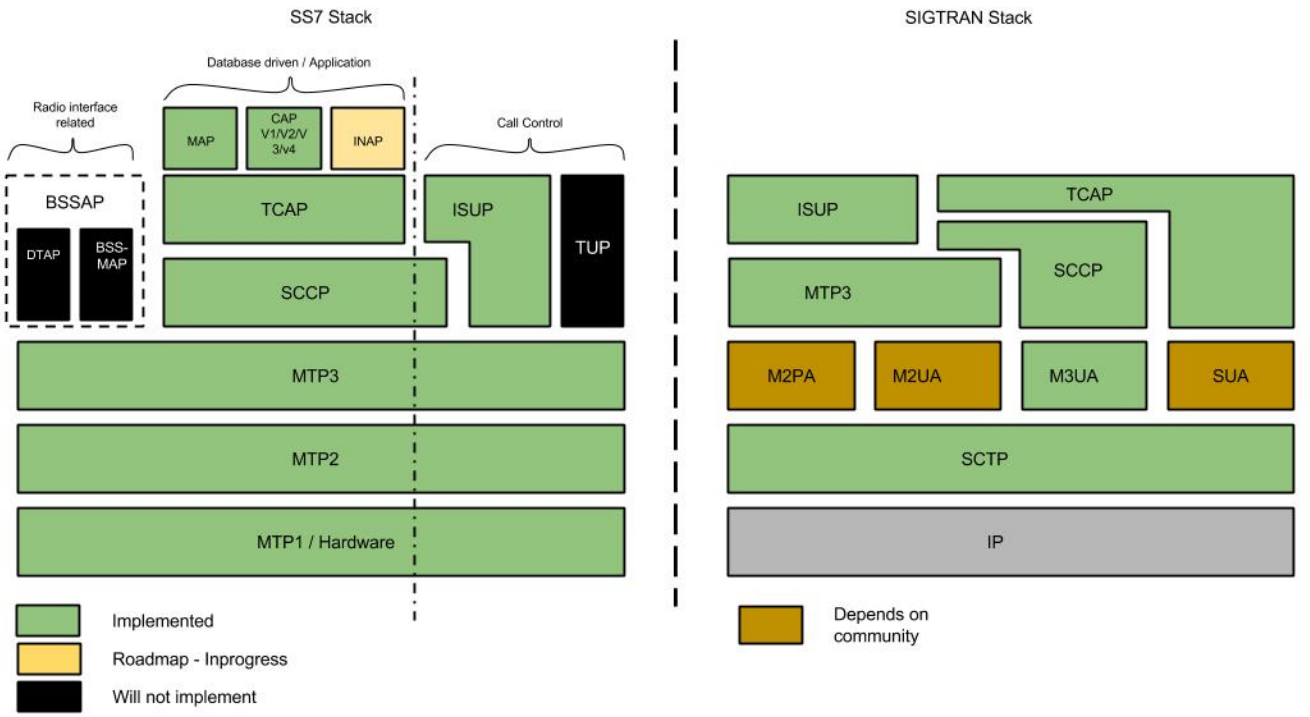
The hardware and software functions of the protocol are divided into functional abstractions called "levels". These levels map loosely to the Open Systems Interconnect (OSI) 7-layer model defined by the International Standards Organization (ISO).

[ss7 fig3] | *images/ss7-fig3.gif*

1.3. Implemented Protocols and Standards Compliance

Restcomm jSS7 is a software based implementation of the SS7 protocol. It provides implementation for Level 2 and above in the SS7 protocol Stack. The Restcomm jSS7 is a platform in the sense that it does not provide the application itself but rather allows users to build their own SS7 applications using Restcomm jSS7 as a platform.

The figure below depicts the various SS7 protocols implemented by Restcomm jSS7 .



Restcomm jSS7 adheres to the standards specified by ITU and ANSI. The table below depicts the implementation standards compliance matrix.

Table 1. Standards Compliance

Stack	**Compliance**
ISUP	ITU-T Q.761 to Q.764 and Q.767
SCCP	ITU-T Q.711 to Q.716, ANSI T1.112-2000

Stack	Compliance
TCAP	ITU-T Q.771 to Q.775, ANSI T1.114-2000
MAP	GSM 09.02, GSM 29.002, GSM 03.40
CAP	GSM 09.78 - CAMEL Phase - I, II, III and IV
M3UA	RFC 4666

M3UA, SCCP and TCAP stacks are compliant with ANSI standard. ANSI support for MAP, CAP and ISUP is not ready.

1.4. Major Features of Restcomm jSS7

Java-based

Restcomm jSS7 is the only Java based SS7 stack. It is robust and reliable and can be installed on any Operating System that supports Java (JDK 7 and SCTP).

Open Source

The Software is open-source, giving you the freedom the understand the code and customise it to your enterprise needs. It is supported by a vibrant Open source community.

SS7 Hardware Cards

Restcomm jSS7 can be used with Intel family boards (Dialogic SS7 cards) or Zaptel/Dahdi compatible TDM devices (Digium, Sangoma).

SIGTRAN (M3UA)

It also has in-built support for SIGTRAN (M3UA using SCTP).Restcomm jSS7 M3UA is based on RFC 4666 and supports ASP, SGW or IPSP modes. It supports both Single Exchange and Double Exchange of messages.

Flexibile and Consistent API

It offers a flexible and consistent API set to develop SS7 applications quickly and efciently irrespective of the lower layer details (legacy SS7 links or SIGTRAN). For example, applications using Restcomm jSS7 SCCP (and/or upper layers) can be easily migrated from TDM equipments to Restcomm jSS7 M3UA with just configuration changes without having to modify a single line of code.

Standalone or JSLEE RA

You can use it as a standalone library or make use of JSLEE RA (Resource Adaptors) that come with it to manage the Stack and develop applications effectively.

Easy Configuration and Management

Restcomm jSS7 comes with an efficient Command Line Interface (CLI) tool allowing you to completely configure the Stack at run-time and manage it using simple commands rather than do everything manually. Restcomm jSS7 also comes with a Graphical User Interface that will allow you to configure, monitor and manage the Stack through a convenient user-friendly interface.

Scalability

Restcomm jSS7 is easily scalable with a configurable load-balancing architecture.

Statistics

Restcomm jSS7 provides teal time statistics at TCAP level indicating number of dialogs, components, error's etc for given time period

Technical Specifications

Restcomm jSS7 is not restricted by any license or Transaction Per Second model. The only restricting factor is memory + CPU capacity of the host servers.

- Restcomm SCTP supports as many associations as supported by the underlying Operating System.
- Restcomm M3UA can be confgured to have as many ASP's / IPSP's as needed by the system.
- Restcomm SCCP can be confgured to have virtually unlimited Global Title Translation rules and also supports wild characters for partial matching of Global Title digits.

Chapter 2. Architecture

2.1. Logical Design

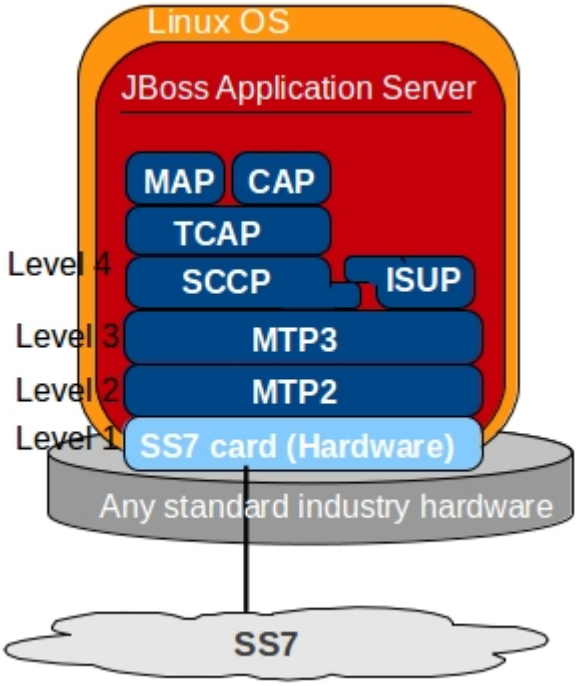
The Restcomm jSS7 is logically divided into two sections - lower and upper.

- The lower section provides implementation for SS7 Level 2 and Level 3. This section is therefore influenced by the type of SS7 hardware (Level 1) used.
- The upper section provides implementation for SS7 Level 4 and above.

This logical division offers great flexibility where hardware is concerned. Irrespective of the type of hardware used in the lower section, Restcomm jSS7 Level 4 and above remains the same. Since the API set is consistent regardless of the lower layers, you can easily migrate your applications from TDM equipments to M3UA. For example, applications using Restcomm SCCP stack (and/or upper layers) can easily be migrated from TDM equipments to Restcomm M3UA with just configuration changes and without changing a single line of code.

Restcomm jSS7 is designed efficiently to offer you the flexibility to install and use the Levels 2,3 and 4 in the same JVM and machine where SS7 Hardware (Level 1) is installed. Alternately, you can also install Level 1,2 and 3 in one machine and install Level 4 on a separate machine. In the second scenario, `M3UA` over `SCTP` is used for communication between Level 3 and Level 4 (on different machines) and this is referred to as Restcomm Signaling Gateway. The figures below illustrate the above 2 scenarios.

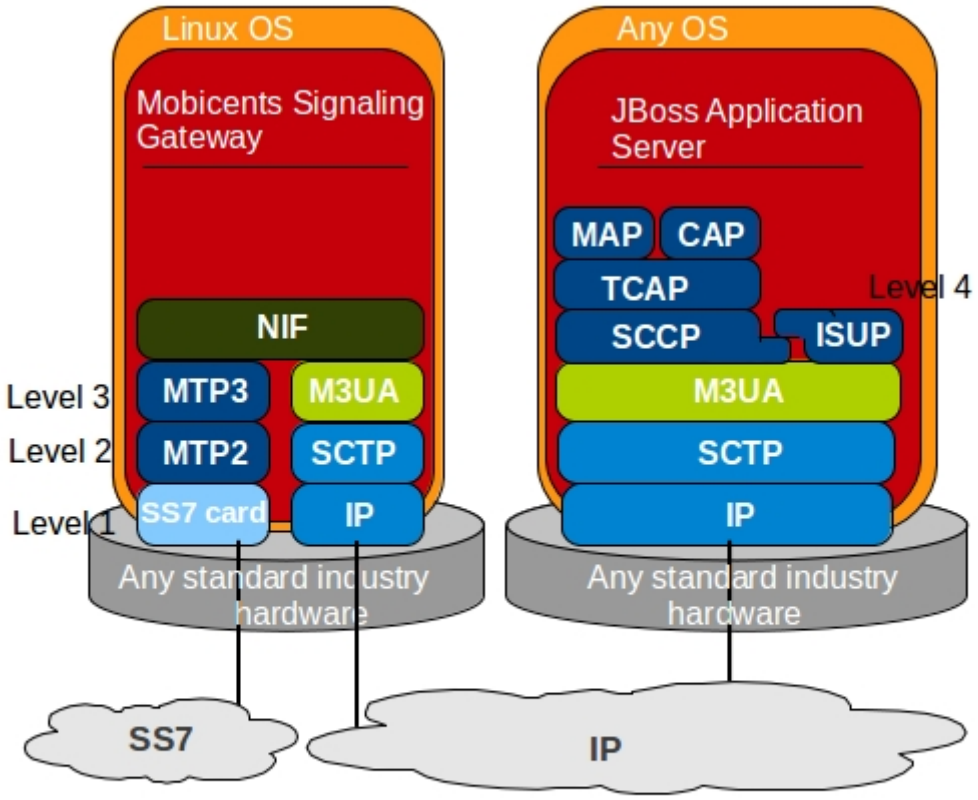
Scenario 1: The complete Restcomm jSS7 is installed in one machine.



The above two sce

Scenario 2: Restcomm Signaling Gateway - Level 3 and below are installed on one machine and

Level 4 is installed on a different machine.



NIF - Nodal Interworking Function

If you use Restcomm M3UA stack, you must use JDK 7 to run the stack as well as to compile the source code. M3UA leverages Java SCTP which is available only in JDK 7.

For more details regarding installation, please refer to the Restcomm jSS7 Installation Guide.

2.2. Directory Structure

The top-level directory is named - and immediately below this are five sub-directories named *asn*, *_docs*, *oam*, *sctp* and *ss7*. All the functional modules of the Stack reside in the *ss7* folder.

```
|- restcomm-jss7-<version>
        |- asn

        |- docs

        |- oam

        |- sctp

        |- ss7
                |+ protocols
                |+ shell
                |+ restcomm-ss7-service
                |+ restcomm-ss7-sgw
                |+ restcomm-ss7-simulator
                |+ restcomm-ss7-traceparser
                |+ template
```

The following is a description of the important services and libraries that make up Restcomm jSS7

asn

Abstract Syntax Notation One (ASN.1) library is used by various Restcomm jSS7 protocols to encode/decode the structured data exchanged between Signaling Points over networks. For more details on the `asn` library, refer to the document included in the *_docs* folder. Applications using any of the Restcomm jSS7 User Protocols may never need to call an `asn` API directly, however it must be in the classpath since Restcomm jSS7 User Protocols refer to this library.

docs

All relevant documentation for Restcomm jSS7 .

oam

UI Management module

sctp

Stream Control Transmission Protocol (SCTP) Library provides the APIs over Java SCTP. This library will be used only if M3UA layer is used. For more details on the `sctp` library, refer to the documentation included in the *docs* folder.

ss7

This folder contains the core protocol libraries that will be used by end applications as well as by the SS7 Service deployed in JBoss AS. The sub-directories included in the *ss7* folder are:

- *restcomm-ss7-sgw* : Standalone Signaling Gateway
- *restcomm-ss7-service* : SS7 service is the core engine and is used in conjunction with JBoss AS. The installation guide will teach you to install the Stack as a standalone component if you do not wish to run it as a JBoss AS Service.
- *restcomm-ss7-simulator* : SS7 Simulator is an application for testing the SS7 stack and

displaying its functionality. It is also a good example of how to use this stack.

- *restcomm-ss7-traceparser* : mobicents jSS7 Stack Trace Parser is a simple tool that can parse trace capture data files (of some formats) and log its content in some log file in a text form
- *protocols* : Restcomm jSS7 User Protocols libraries. Your application will directly call the APIs exposed by these libraries. Depending on the application, you may be interested in either TCAP or MAP, or both, or ISUP libraries.
- *shell* : This holds the Command Line Interface (CLI) module to manage the Restcomm jSS7 .
- *template* : This folder contains templates that are needed for JSS7 stack configuring.

2.3. Functional Blocks

The major functional modules of the jSS7 are:

1. SS7 Service *[dir: restcomm-ss7-service]*
2. Signaling Gateway *[dir: restcomm-ss7-sgw]*
3. Shell *[dir: shell]*
4. GUI *[dir: ui]*
5. SS7 Simulator *restcomm-ss7-simulator]*

The following sub-sections discuss in detail about the functionality of these individual components.

2.3.1. SS7 Service

SS7 Service creates an instance of higher layer of the Restcomm Stack and binds the instance to JNDI. SS7 Service is a JMX based service deployed in JBoss Application Server. It hides the underlying details like whether Level 4 and above are connected to peer via M3UA or if connected to the SS7 Hardware installed in the same machine as Level 4.

Following services are bound:

Table 2. SS7 Services

Stack Name	JNDI Name	Comments
TCAP	java:/restcomm/ss7/tcap	Exposes TCAP Stack via JNDI
MAP	java:/restcomm/ss7/map Exposes	MAP Stack via JNDI
CAP	java:/restcomm/ss7/cap Exposes	CAP Stack via JNDI
ISUP	java:/restcomm/ss7/isup	Exposes ISUP stack via JNDI

The figure below depicts the elements that are deployed as part of the SS7 MAP Service.

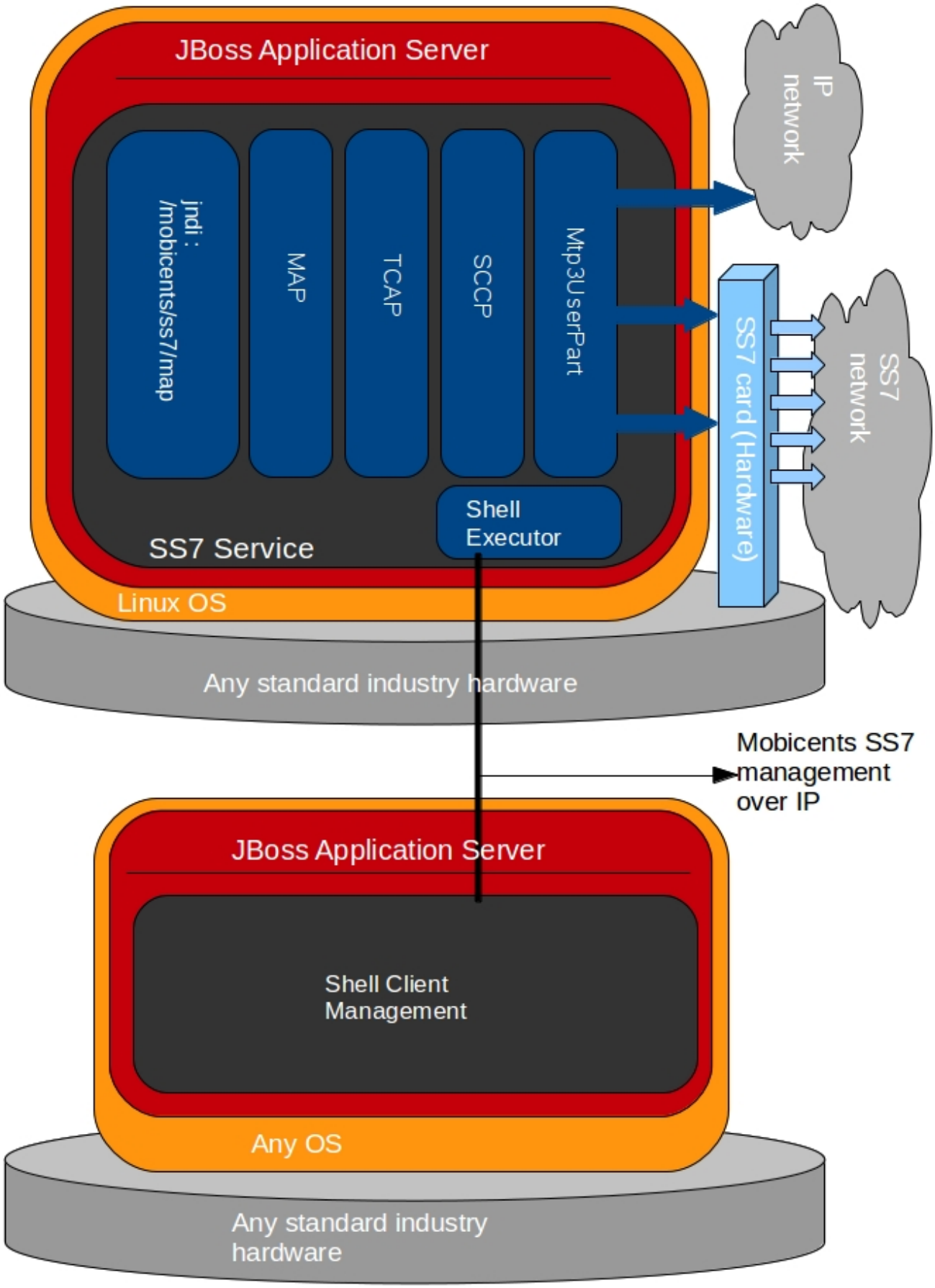


Figure 1. Restcomm jSS7 Stack Service Elements

SS7 Service Elements serve the following purposes:

Expose protocol access points

Access points allow users to access lower layer protocols like `MAP` and interact with the SS7 network through such protocols.

Expose management interface

`Shell Executor` allows the `Shell` client to connect and issue commands.

The figure below depicts the elements that are deployed as part of SS7 Service.

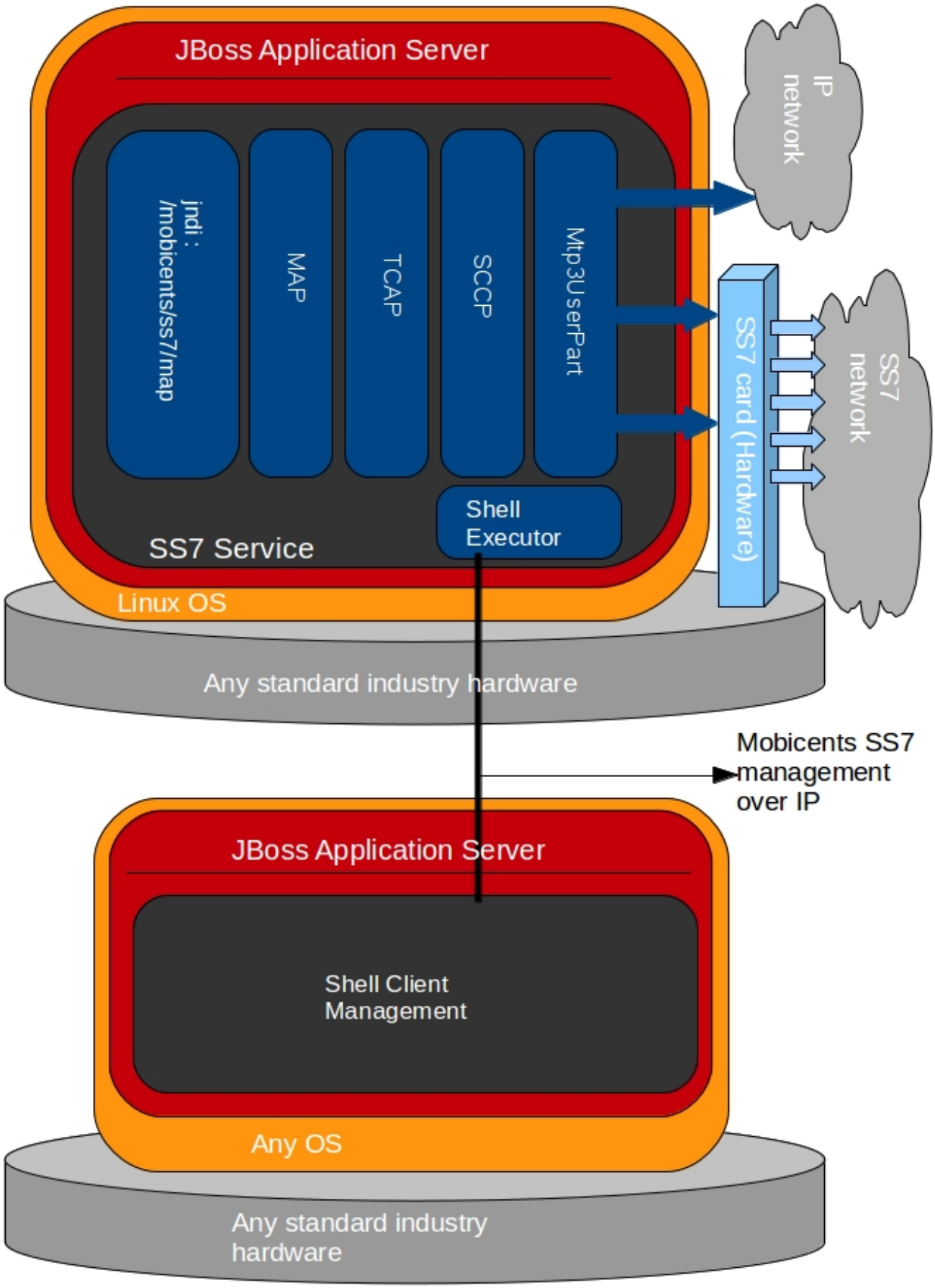


Figure 2. Restcomm jSS7 Stack Service Elements

For more details on Running and Configuring the SS7 Service Elements, please refer to the chapter Running.

Stack Usage

The figure below depicts how Restcomm jSS7 is used.

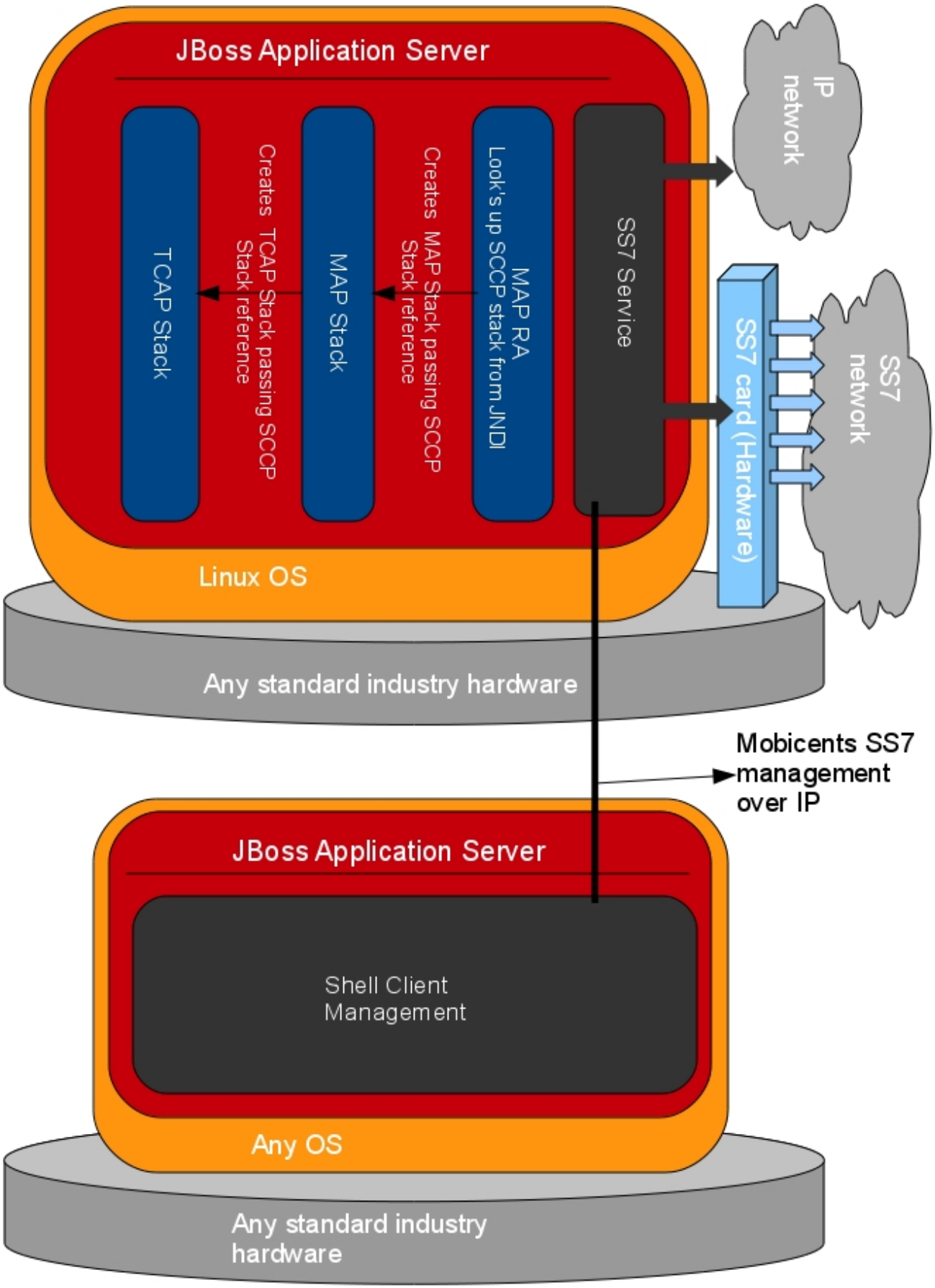


Figure 3. Restcomm jSS7 Stack General Design

2.3.2. Signaling Gateway

Restcomm Signaling Gateway (SG) is a signaling agent that receives and sends Switched Circuit Network (SCN) native signaling at the edge of the IP network. Restcomm Signaling Gateway leverages MTP and Restcomm `M3UA` Stack explained in Restcomm Signaling Gateway M3UA Stack.

The figure below shows the components included in Restcomm Signaling Gateway. Configuring the Signaling Gateway is explained in the chapter Running.

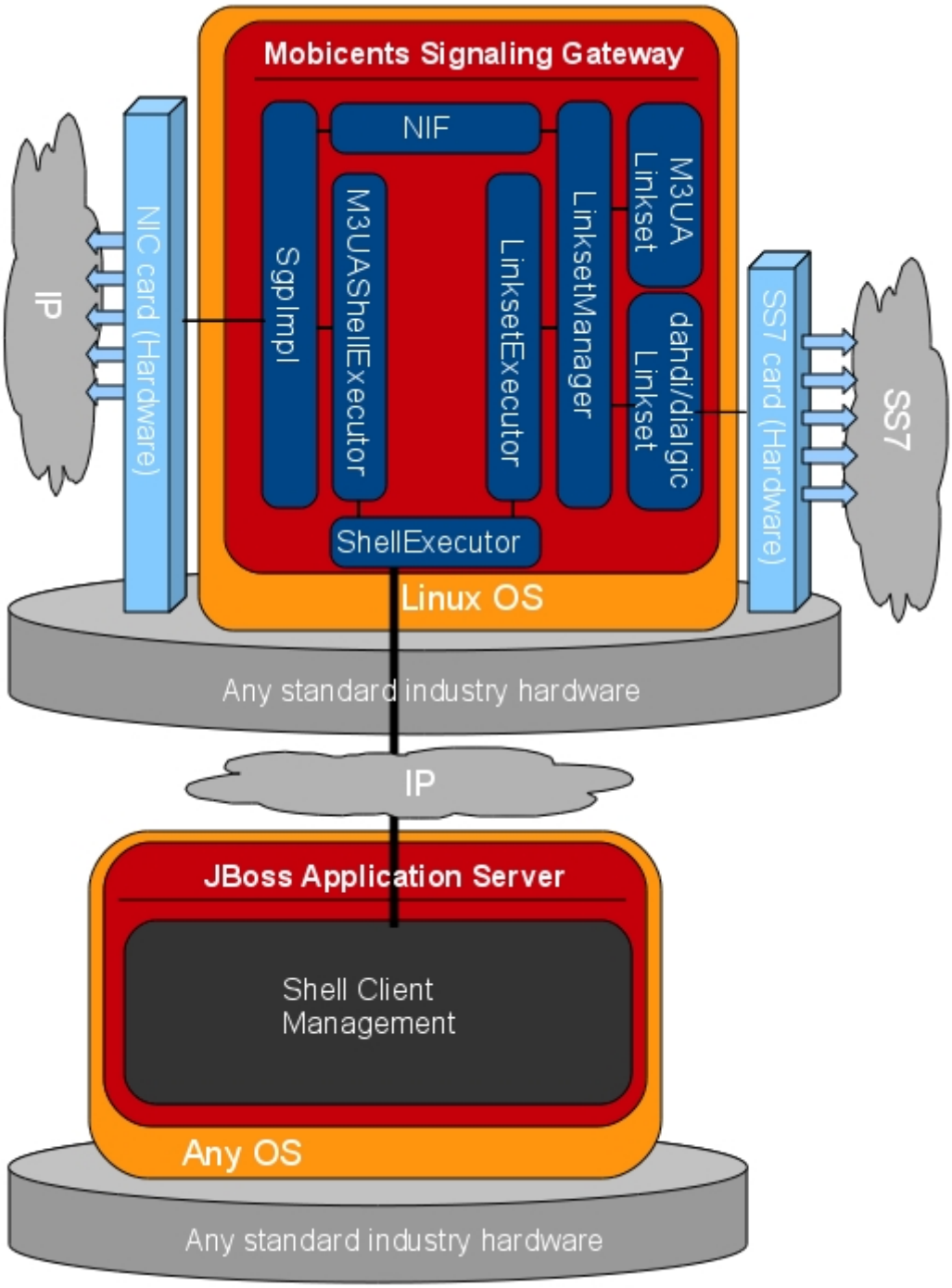


Figure 4. Restcomm Signaling Gateway Components

2.3.3. Shell - Comman Line Interface

`Shell` is a Command Line Interface (CLI) tool that will allow you to manage different aspects of

Restcomm jSS7 in an interactive manner. It connects to different instances of Restcomm jSS7 which manages `Linksets`, `SCCP` resource, routing and `M3UA`.Usually `Shell` will be invoked from a remote machine(remote to `Linksets` and application protocols).

2.3.4. Graphical User Interface

The Graphical User Interface will allow you to manage different aspects of Restcomm jSS7 through a convenient user-friendly interface. You can launch the GUI in any Web Browser and manage the Stack instance efficiently using the GUI operations.

2.3.5. SS7 Simulator

Restcomm jSS7 comes with a Simulator that will help you to understand the functionality of the Stack. The Simulator may be used as an application for testing the SS7 Stack or as an example of how to use this Stack. You can run several instances of the Simulator in a single machine and each instance can have its own configuration. In addition, the Simulator offers you the flexibility to run it locally or remotely. You must remember to configure all layers before running tests with the Simulator.

The Simulator contains three layers of SS7 stack components and one testing task layer which presents the concrete testing task. You can select from these layers as required, however some layers demand corresponding lower layers. For example, the `TCAP+MAP` layer demands `SCCP` as layer 2. Depending on your testing goals, you can configure each of these layers separately but the configuration options are limited and do not cover all possible SS7 Stack options.

Simulator Layers

1. Layer 1 [MTP3]
    - M3UA
    - DialogicCard
    - DahdiCard [yet to be implemented]
2. Layer 2
    - SCCP
    - ISUP [yet to be implemented]
3. Layer 3
    - TCAP + MAP
    - TCAP + CAP
    - TCAP + INAP [yet to be implemented]
4. Testing Task Layer
    - USSD client test
    - USSD server test
    - SMS server test

- SMS client test
- CAP SSF test
- CAP SCF test
- MAP ATI client test
- MAP ATI server test
- MAP Check IMEI client test
- MAP Check IMEI server test

Chapter 3. Multi tenancy support

Multi tenancy allows same instance of Restcomm jSS7 to connect to different operators, each having its own links and point-codes.

Multi tenancy is achieved by having a model of SS7 network as a splitted to several logical networks. Each logical network has a corresponded key "networkId". It is digital one and default value (if we do not specify it) - 0.

NetworkId value is assigned for a configurable objects:

- SCCP SAP
- SCCP Rule

NetworkId is also assigned to activities:

- SccpMessage (assigned by Service Acess Point - SAP for SS7 originated messages, assigned by TCAP Dialog for TCAP originated messages)
- TCAP Dialog (assigned by SccpMessage for SCCP originated messages, assigned by upper Dialog for user messages)
- MAP / CAP Dialog (same as TCAP Dialog)
- SCCP rule is taken into account only when SCCP rule networkId == SccpMessage networkId

Chapter 4. Running

You must ensure that you follow the steps outlined in the Restcomm jSS7 Installation Guide to install the Components. As explained in the Installation Guide, you may install jSS7 as a standalone library or as a JBoss AS SS7 Service or as a Signaling Gateway. Once installed, you can run the different Components as described in the sections below.

4.1. Running Restcomm SS7 Service

Procedure: Start SS7 Service

1. Pre-requisite: You must have SS7 Service deployed in the JBoss Application Server as explained in the Installation Guide.
2. All you have to do to start the Service is start the JBoss AS. This will automatically start the SS7 Service. To start the JBoss Server you must execute the *run.sh* (Unix) or *run.bat* (Microsoft Windows) startup script in the *<jboss_install_directory>/bin* folder (on Unix or Windows).
3. Result: If the service started properly you should see following lines in the Unix terminal or Command Prompt depending on your environment:

```
14:07:47,580 INFO  [TomcatDeployment] deploy, ctxPath=/admin-console
14:07:47,650 INFO  [config] Initializing Mojarra (1.2_12-b01-FCS) for context
'/admin-console'
14:07:49,980 INFO  [TomcatDeployment] deploy, ctxPath=/
14:07:50,032 INFO  [TomcatDeployment] deploy, ctxPath=/jmx-console
14:07:50,235 INFO  [ManagementImpl] SCTP configuration file path /home/vinu/jboss-
5.1.0.GA/server/default/data/SCTPManagement_sctp.xml
14:07:50,262 INFO  [ManagementImpl] Started SCTP Management=SCTPManagement
WorkerThreads=0 SingleThread=true
14:07:50,269 INFO  [SelectorThread] SelectorThread for Management=SCTPManagement
started.
14:07:50,422 INFO  [M3UAManagement] M3UA configuration file path /home/vinu/jboss-
5.1.0.GA/server/default/data/Mtp3UserPart_m3ua.xml
14:07:50,512 INFO  [M3UAManagement] Started M3UAManagement
14:07:50,605 INFO  [SccpStackImpl-SccpStack] Starting ...
14:07:50,665 INFO  [Router] SCCP Router configuration file path /home/vinu/jboss-
5.1.0.GA/server/default/data/SccpStack_sccprouter.xml
14:07:50,667 INFO  [Router] Started SCCP Router
14:07:50,668 INFO  [SccpResource] SCCP Resource configuration file path
/home/vinu/jboss-5.1.0.GA/server/default/data/SccpStack_sccpresource.xml
14:07:50,669 INFO  [SccpResource] Started Sccp Resource
14:07:50,671 INFO  [SccpStackImpl-SccpStack] Starting routing engine...
14:07:50,671 INFO  [SccpStackImpl-SccpStack] Starting management ...
14:07:50,671 INFO  [SccpStackImpl-SccpStack] Starting MSU handler...
14:07:50,696 INFO  [ShellExecutor] Starting SS7 management shell environment
14:07:50,714 INFO  [ShellExecutor] ShellExecutor listening at /127.0.0.1:3435
14:07:50,776 INFO  [TCAPStackImpl] Starting
...org.mobicents.protocols.ss7.tcap.TCAPProviderImpl@14d18a1
14:07:50,776 INFO  [TCAPProviderImpl] Starting TCAP Provider
14:07:50,776 INFO  [TCAPProviderImpl] Registered SCCP listener with address 8
14:07:51,012 INFO  [SS7Service]  [[[[[[[[[ Mobicents jSS7 3.0.0 service started
]]]]]]]]]
14:07:51,156 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-127.0.0.1-8080
14:07:51,177 INFO  [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
14:07:51,184 INFO  [ServerImpl] JBoss (Microcontainer) [5.1.0.GA (build:
SVNTag=JBoss_5_1_0_GA date=200905221053)] Started in 40s:691ms
```

4. If you are starting Restcomm-jSS7-7.0.0-SNAPSHOT for the first time, SS7 is not configured. You need to use the Shell Client to connect to Restcomm-jSS7-7.0.0-SNAPSHOT . Using CLI you can configure how service interacts with SS7 network, that is you configure either installed SS7 card and its native library , or `M3UA` layer. You can also use the GUI to achieve the same. Once configured, the state and configuration of SS7 is persisted which stands server re-start.

Procedure: Stop SS7 Service

1. To stop the SS7 service, you must shut down the JBoss Application Server. To shut down the server(s) you must execute the `shutdown.sh -s` (Unix) or `shutdown.bat -s` (Microsoft Windows) script in the *<jboss_install_directory>/bin* directory (on Unix or Windows).
2. If the Service stopped properly, you will see the following three lines as the last output in the

Unix terminal or Command Prompt:

```
[Server] Shutdown complete
Halting VM
```

4.2. Running Restcomm Signaling Gateway

4.2.1. Start the Gateway

Procedure: Start the Restcomm Signaling Gateway on

1. Change the working directory to the installation directory (the one into which the zip file's contents was extracted to)

   ```
   downloads]$ cd restcomm-jss7-<version>/ss7/restcomm-ss7-sgw
   ```

2. (Optional) Ensure that the *bin/run.sh*start script is executable.

   ```
   restcomm-ss7-sgw$ chmod +x bin/run.sh
   ```

3. Execute the *run.sh*Bourne shell script.

   ```
   restcomm-ss7-sgw$ ./bin/run.sh
   ```

4. In the Linux terminal, the Restcomm Signaling Gateway has started successfully if the last line of output is similar to the following

   ```
   15:51:18,247 INFO  [MainDeployer] [[[[[[[[[ Mobicents Signaling Gateway:
   release.version= Started ]]]]]]]]]
   ```

Procedure: Start the Restcomm Signaling Gateway on

1. Using Windows Explorer, navigate to the *restcomm-jss7-<version>/ss7/restcomm-ss7-sgw/bin* subfolder in the installation directory.
2. The preferred way to start the Restcomm Signaling Gateway is from the Command Prompt. The command line interface displays details of the startup process, including any problems encountered during the startup process.

   Open the Command Prompt via the Start menu and navigate to the correct folder:

   ```
   C:\Users\<user>\My  Downloads>cd "restcomm-jss7-<version>\ss7{this-folder}-ss7-sgw"
   ```

3. Start the Gateway by executing the *run.bat* file:

- *run.bat* batch file:

```
C:\Users\<user>\My Downloads\mms-standalone<version>>bin\run.bat
```

4.2.2. Stop the Gateway

The only option to stop the gateway is by pressing `Ctrl c` and bringing down the JVM or killing the process.

4.2.3. Bind the Gateway to IP

Using `run.sh` without any arguments binds the gateway to `127.0.0.1`. To bind the Gateway to a different IP, pass the IP address as value to the `-b` command line option. For example to bind the server to `115.252.103.220` you will use the command as below:

```
restcomm-ss7-sgw$ ./bin/run.sh -b 115.252.103.220
```

4.3. Running jSS7 as a Standalone library

If you do not wish to use the JBoss Container and do not require JSLEE RAs you can configure and run the Stack as a Standalone library. For more details, refer to Configuring as a Standalone library

4.4. Running the Shell`Shell`

4.4.1. Start the Shell Client

Shell client can be started with following command from *$JBOSS_HOME/bin*:

```
[$] ./ss7-cli.sh
```

Once console starts, it will print following like information:

```
=========================================================================

Mobicents SS7 Management Shell Bootstrap Environment

=========================================================================

mobicents>
```

The `ss7-cli`script supports the following options

```
Usage: SS7 [OPTIONS]
Valid Options
-v           Display version number and exit
-h           This help screen
```

4.4.2. Connect to Managed Instance

Shell needs to connect to managed instance. Command to connect has following structure:

```
connect <IP> <PORT>
```

Example 1. Connect to remote machine

```
mobicents>connect 10.65.208.215 3435

mobicents(10.65.208.215:3435)>
```

Host IP and port are optional, if not specified, shell will try to connect to `127.0.0.1:3435`

4.4.3. Disconnect

Command to disconnect has following structure:

```
ss7 discconnect
```

Example 2. Disconnect

```
mobicents(10.65.208.215:3435)>ss7 disconnect

Bye
mobicents>
```

4.4.4. Authentication and Audit Logs

Security

Security is a fundamental requirement of any Telecom application. You must control access to your SS7 network and restrict who is allowed to access what and perform what operations.

Restcomm jSS7 CLI Security is based on the JBoss Security Framework. The JBoss Security framework provides support for a role-based declarative security model as well as integration of

custom security via a security proxy layer. The default implementation of the declarative security model is based on Java Authentication and Authorization Service (JAAS) login modules and subjects.

Procedure: Enable Security

1. Add a new parameter named “securityDomain” to the "ShellExecutor" bean in the configuration file *jboss-5.1.0.GA/server/default/deploy/restcomm-ss7-service/META-INF/jboss-beans.xml* and save the changes.

```
<property name="securityDomain">java:/jaas/jmx-console</property>
```

2. Configure the security domain in the file *jboss-5.1.0.GA/server/default/conf/login-config.xml* following the instructions in the JBoss Admin Guide.
3. Create entries for user id and password in the file [path]_ jboss-5.1.0.GA/server/default/conf/props/jmx-console-users.properties_ for every user allowed to access the CLI.

Procedure: Disable Security

1. Delete all configurations created as mentioned above and remove the parameter “securityDomain” from the "Shell Executor" bean defined in *jboss-5.1.0.GA/server/default/deploy/restcomm-ss7-service/META-INF/jboss-beans.xml*.

If you would like to read more about the JBoss Security Framework, please refer to the JBoss Admin Guide available in their website.

Audit Configuration

If security is enabled then you can log the operations performed by every user.

Procedure: Enable Audit

1. Add a new appender to the file *jboss-5.1.0.GA/server/default/conf/jboss-log4j.xml* as below:

```
<appender name="AUDIT" class="org.jboss.logging.appender.DailyRollingFileAppender">
    <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="File" value="${jboss.server.log.dir}/audit.log"/>
    <param name="Append" value="true"/>
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d %-5p [%c] (%t:%x) %m%n"/>
    </layout>
</appender>
```

2. Add a new category to the file *jboss-5.1.0.GA/server/default/conf/jboss-log4j.xml* as below:

```
<category name="org.jboss.security.audit.providers.LogAuditProvider"
additivity="false">
    <priority value="TRACE"/>
    <appender-ref ref="AUDIT"/>
</category>
```

When security and audit is enabled, a sample audit log looks like this:

```
2012-11-28 22:17:27,005 TRACE [org.jboss.security.audit.providers.LogAuditProvider]
(pool-19-thread-8:) [Success]message=login success;principal=admin;
2012-11-28 22:17:47,659 TRACE [org.jboss.security.audit.providers.LogAuditProvider]
(pool-19-thread-1:) [Info]response=Successfully added client
Association=SCTPAssoc1;principal=admin;command=sctp association create SCTPAssoc1
CLIENT 127.0.0.1 2775 127.0.0.1 2776;
2012-11-28 22:18:06,773 TRACE [org.jboss.security.audit.providers.LogAuditProvider]
(pool-19-thread-3:) [Success]message=logout success;principal=admin;
```

4.4.5. Command-line Completion

Restcomm jSS7 Shell supports Command-line completion (Tab Completion) allowing you to type the first (or first few) character(s) of the comand and press tab to fill in the rest of the command. As soon as you enter the CLI (by executing the *ss7-cli.sh* script) you can make use of this feature to view all possible commands.

When you first enter CLI and press the "tab" key, CLI will display all operations permitted in that context. Once you connect to a managed instance and while staying connected if you press the "tab" key it will display all other commands allowed.

```
mobicents> [tab key press]
history  connect  exit
mobicents>connect 10.65.208.215 3435
mobicents(10.65.208.215:3435)> [tab key press]
sctp        linkset     m3ua        sccp        history     disconnect
```

If you enter the first few characters of a command and press "tab", CLI will automatically fill in the rest of the command or display all possible commands if there is more than one command beginning with the characters entered by you.

```
mobicents(10.65.208.215:3435)>sctp [tab key press]
server       association  --help
mobicents(10.65.208.215:3435)>sctp
```

In addition, help files are also available for every command using the --help option. The help files provide details of the command including possible parameters and examples of usage if applicable.

```
mobicents(10.65.208.215:3435)>sctp --help
Name
    sctp
    Manage M3UA - SCTP

SYNOPSIS
    sctp server [create | destroy | start | stop | show]  [parameters]

    sctp association [create | destroy | show]  [parameters]

    parameters
        Command Line parameters.

DESCRIPTION
    This command is used to manage M3UA - SCTP. You can create, destroy, start and
    stop a SCTP Server and view the Server configuration using the sctp server
    command. You can create, destroy and view SCTP Associations using the
    sctp association command.

SEE ALSO
    sctp server create, sctp server destroy, sctp server start, sctp server stop,
    sctp server show, sctp association create, sctp association destroy,
    sctp association show

mobicents(10.65.208.215:3435)>
```

4.5. Running the Graphical User Interface

Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. The window will look similar to the figure below. The GUI is divided into three sections:

- A left panel listing the management units (Services, SCTP, M3UA, Linkset, SCCP, TCAP, Alarms, Manage Campaigns and Metrics). You can click on any of these to select and navigate to the specific management unit.
- A main panel displaying the currently selected management unit. At the top of this panel you will find a bread crumb trail providing links back to each previous page that you navigated through in order to get to the current page. The main view is categorized into multiple tabs to manage different aspects of the selected layer.
- A bottom panel displaying the log data. You can clear the log anytime by clicking on the trash icon at the top right corner of this panel. You can also minimize or maximize this panel to suit your needs.

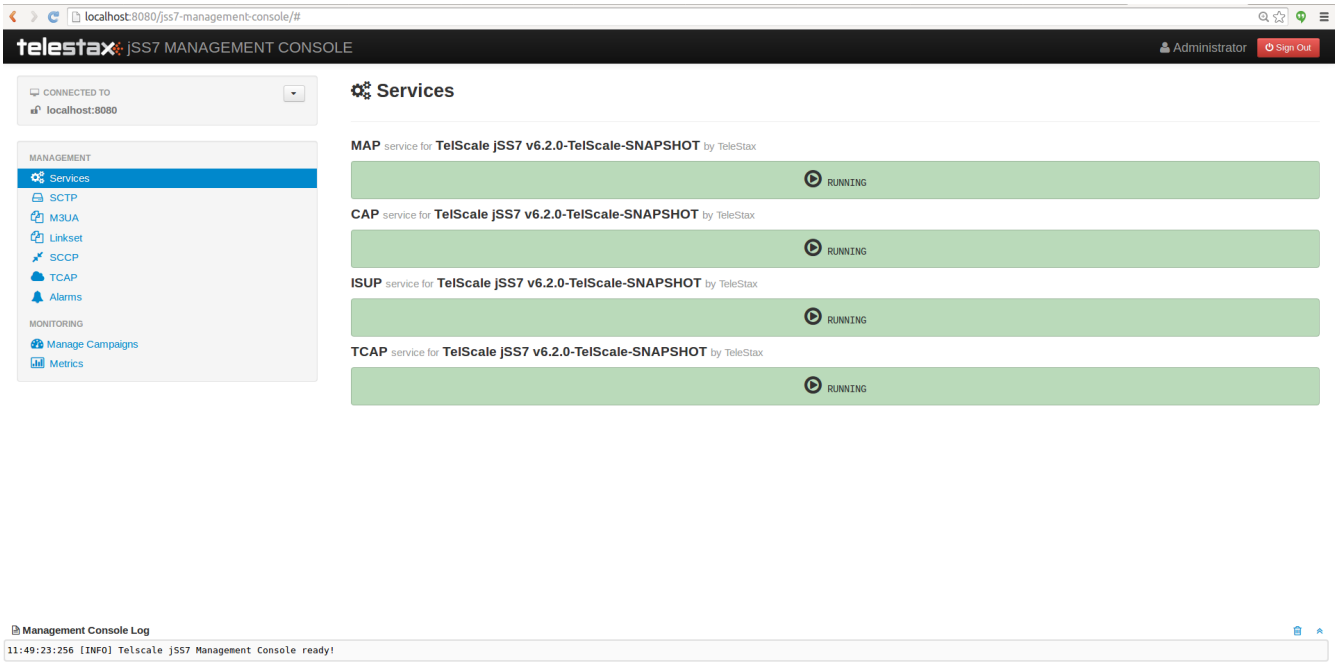


Figure 5. GUI - Services

The main window will display all the configured services and their current state (running or stopped). In the example above MAP, CAP, TCAP and ISUP are configured and running. If any of these services is not running currently, the page will indicate that service as 'Stopped'. This page only indicates the current state of a service and does not allow you to start or stop the service dynamically. You can view this screen anytime by clicking on the 'Services' link in the left panel.

You will notice bread crumbs at the top of the main panel, allowing you to return to any of the previous pages you navigated through.

4.5.1. Connect to a new Instance

You can connect to a new instance by entering the IP:Port values and the login credentials in the top left corner of the GUI. However please note that this feature is not available in this release but will be fully functional in the next release.

4.5.2. Authentication

Restcomm jSS7 GUI Management Security is based on the JBoss Security Framework. This is explained in [_security].

4.6. Running Restcomm jSS7 Simulator

The Restcomm jSS7 comes with a Simulator module that will enable you to test and understand the functionality of the Stack. You can install and run the Simulator module on any machine. In addition, you can run several instances of the Simulator from one folder and each of these instances can have it’s own configuration options. Configuration options are saved into a xml configuration file.

Each running instance of the Simulator has its own name (called "host name") and the name of configuration file depends on this "host name". For example if the host name is "a1", then the name

of the configuration file will be "a1_simulator.xml". You must ensure that you provide each running Simulator instance with a unique host name in order to allow each running Simulator instance to have different configuration options.

You can run and manage the Simulator locally or remotely. For running the Simulator locally you will be using a GUI interface. If you are intending to run the Simulator remotely, then you can do so using RMI access (via a GUI interface) or a HTML interface (using a HTML Browser).

4.6.1. Running the Simulator Locally

Procedure: Launching the Simulator Locally

1. Pre-requisite: You must have SS7 Simulator installed locally in the machine.
2. Change the working directory to the bin folder in the Simulator's installation directory.

```
downloads]$ cd restcomm-jss7-<version>/ss7/restcomm-ss7-simulator/bin
```

3. (Optional) Ensure that the *run.sh*start script is executable.

```
bin$ chmod +x run.sh
```

4. Execute the *run.sh*Bourne shell script with the command `./run.sh gui`. If you wish to pass the "host name" parameter (say for example "a1") you can do so by issuing the command as `./run.sh gui -na1` or `./run.sh gui --name=a1` where "a1" is the "host name" of this simulator instance.

```
bin$ ./run.sh gui --name=a1
```

5. Result: This will launch the GUI application form ""Connecting to a testerHost ..." as in the figure below:

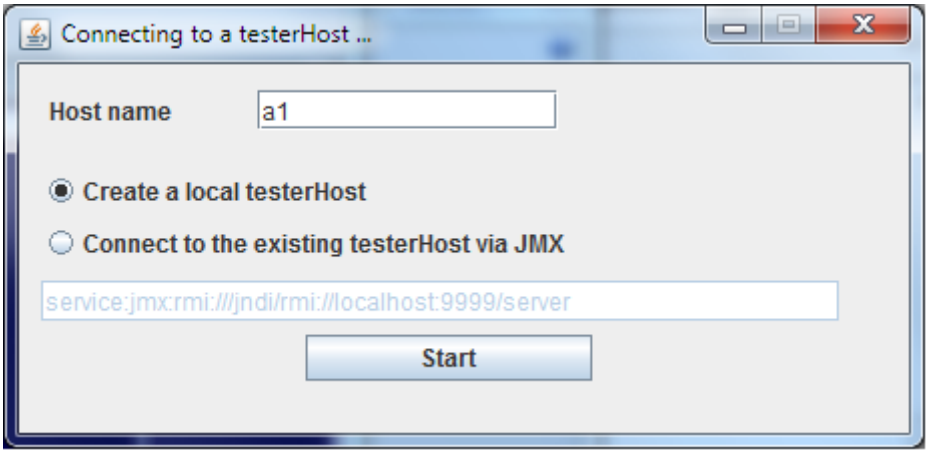


Figure 6. Running Simulator Locally

Procedure: Creating a Local testerHost

1. Pre-requisite: You must have the GUI application form "Connecting to a testerHost" launched by

running the simulator locally as explained in the above procedure.

2. In the GUI application form "Connecting to a testerHost ..." enter a value for the Host name field if it is empty.
3. Select the option "Create a local testerHost" and press the "Start" button.
4. Result: This will create a local testerHost and launch a new form **SS7 Simulator: <host name | -local** as in the figure below:

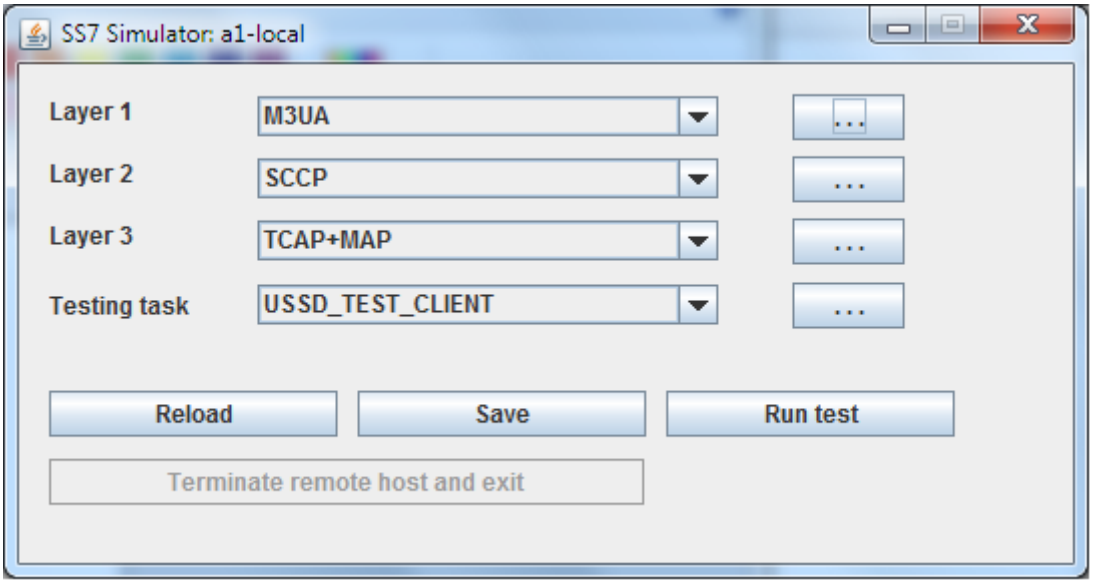


Figure 7. The main form of the Simulator

Procedure: Running a Test

1. Pre-requisite: You must have the main Simulator form launched as explained in the above procedure.
2. Select required modules for Layers 1 - 3 and the Testing Task.
3. Configure the modules to meet your requirements. Click on the button "..." to the right of the selected module. When layer configuring you can press one of two buttons "Load default values for side A" and "Load default values for side B" for loading default values for testing. These default values can be used if you are using SS7 Simulators in the one computer host as the side A and the side B for interaction. To save the configured layers to the disk, click the "Save" button. To reload the saved configuration, use the "Reload" button.
4. When all required layers are correctly configured, click the "Run test" button.
5. Result: The form for testing will be displayed as in the figure below:

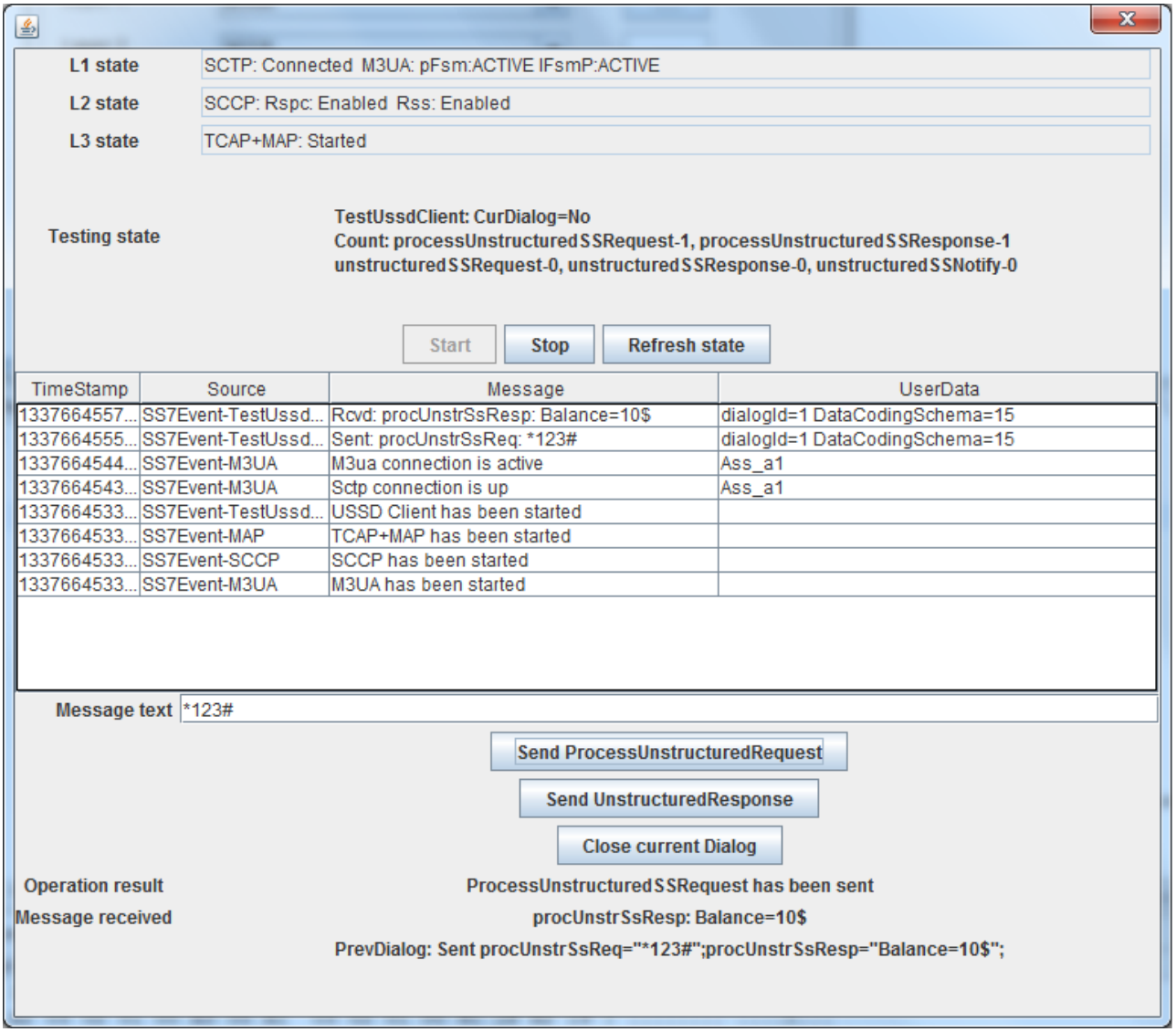


Figure 8. Example Simulator testing form

The form has three sections:

- The top section displays information regarding the Simulator layers and the state of Testing. This information is refreshed every 5 seconds automatically. You can also choose to refresh this manually by clicking the "Refresh state" button. When you click the "Start" button, all modules will start and be ready for testing. When you click the "Stop" button all modules will stop.
- The middle section of the form displays notifications from testing modules.
- The bottom section of the form holds test dependant information whose details are displayed in the corresponding test definition.

For more details on executing tests, please refer to SS7 Simulator Test Cases.

4.6.2. Running SS7 Simulator remotely

The Simulator can be run remotely using RMI and/or HTML Adaptors. But prior to launching the Simulator remotely, you must first run a tester host in the host machine as explained in the procedure below:

Procedure: Launching a Tester Host in the host machine

1. Pre-requisite: You must have SS7 Simulator installed locally in the host machine.
2. Change the working directory to the bin folder in the Simulator's installation directory.

```
downloads]$ cd restcomm-jss7-<version>/ss7/restcomm-ss7-simulator/bin
```

3. (Optional) Ensure that the *run.sh* start script is executable.

```
bin$ chmod +x run.sh
```

4. Execute the *run.sh* Bourne shell script with the command `./run.sh core -na1 -t8001 -r9999,9998` or `./run.sh core --name=a1 --http=8001 --rmi=9999,9998`. You can use a `rmi` connector or `http` connector or both simultaneously. The command options are explained below:
   - "-n" (or "--name=") defines a host name (in the example above: "a1").
   - "-r" (or "--rmi=") defines a port's listening for rmi requests (in the example above: "9999 and 9998"). RMI protocol uses 2 ports for server access. The first port (9999 above) is rmi port and 9998 is second port used by RMI. Usually RMI randomly selects this 2nd port, however if the core simulator is behind firewall, the GUI will not be able to connect from remote machine without admin opening up all the ports. This can be risky, hence --rmi takes two ports as comma separated and admin can open these two ports on firewall.

Please note that fixing the 2nd port in RMI contradicts the RMI specification.

   - "-t" (or "--http=") defines a port listening for html requests (in the example above: "8001").
   - "-Djava.rmi.server.hostname=192.168.1.1" - if remote host contains several IP addresses you need to specify by this command which exactly IP address RMI will use.

```
bin$ ./run.sh core -na1 -t8001 -r9999,9998
```

   or

```
bin$ ./run.sh core --name=a1 --http=8001 --rmi=9999,9998
```

5. Result: This will start the Tester Host in this machine and the output displayed in the console will be as below:

```
=========================================================================

  SS7 Simulator Bootstrap Environment

  SIMULATOR_HOME: /home/vinu/restcomm-jss7-6.1.3.GA/ss7/restcomm-ss7-simulator

  JAVA: /usr/lib/jvm/jre-1.7.0-openjdk/bin/java

  JAVA_OPTS: -Dprogram.name=run.sh  -Djava.net.preferIPv4Stack=true -Xms256m
-Xmx512m -Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000

  CLASSPATH: /home/vinu/restcomm-jss7-6.1.3.GA/ss7/restcomm-ss7-
simulator/bin/run.jar

=========================================================================
.
.
.
.
.
All beans have been loaded...
RMI connector initializing...
RMI connector has been started...
Html connector initializing...
Html connector has been started...
Waiting for commands...
```

When the Tester Host is running successfully in the host machine, you can now run the Simulator tests remotely from any machine which has the Simulator installed. The following procedures provide instructions to run the Simulator remotely using the two interfaces (RMI and HTML).

Procedure: Managing the Simulator Remotely using RMI Interface

1. Pre-Requisite: The Tester Host must be launched in the host machine as explained in the above procedure. The local machine must have the Simulator installed.
2. The best client for managing the Simulator via a RMI interface is the GUI interface of Simulator. It can be launched in the same way as launching a Simulator locally:

   ```
   ./run.sh gui
   ```

   This will launch the GUI form "Connecting to a testerHost...".

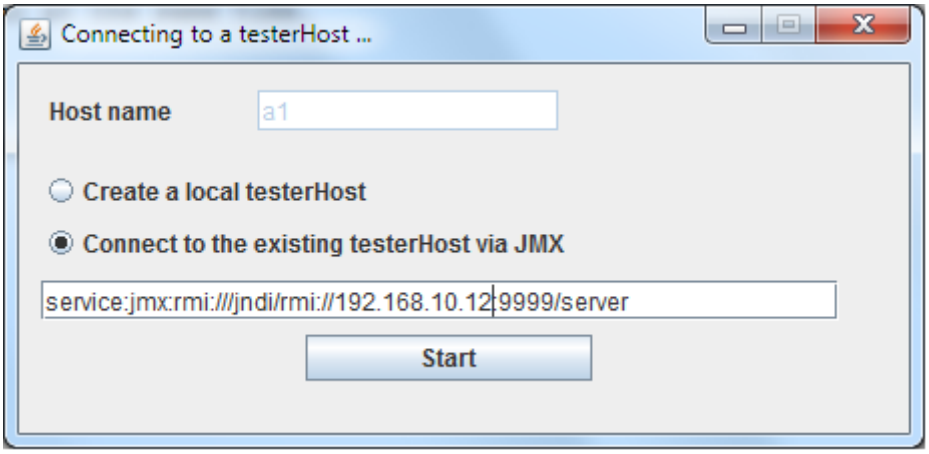


Figure 9. Launching Simulator GUI for a remote mode

3. In the form, select the option "Connect to the existing testerHost via JMX".
4. In the default RMI connection string, replace "localhost" with the correct IP address of the tester host computer and the value "9999" with the correct RMI port (the port from option "-r" or "--rmi=" as provided when launching the testerHost in the host machine).
5. Once successfully connected, the management will be similar to the procedures explained for running the Simulator application locally (see Running the Simulator Locally). "Jconsole" application can be also used as the client. But this application is less convenient when compared to the easy RMI interface.

Procedure: Managing the Simulator Remotely using HTML Interface

1. HTML management is less convenient than RMI management. But it can be used if RMI is not acceptable (for example if you are behind a proxy). For configuring the parameters and running the tests you can use any HTML browser. In the URL field of the browser, type **http://<IP address | :<port** where IP address is the host machine's address and port is the value specified for "-t" or "--http=" option when launching the Tester Host in the remote machine.

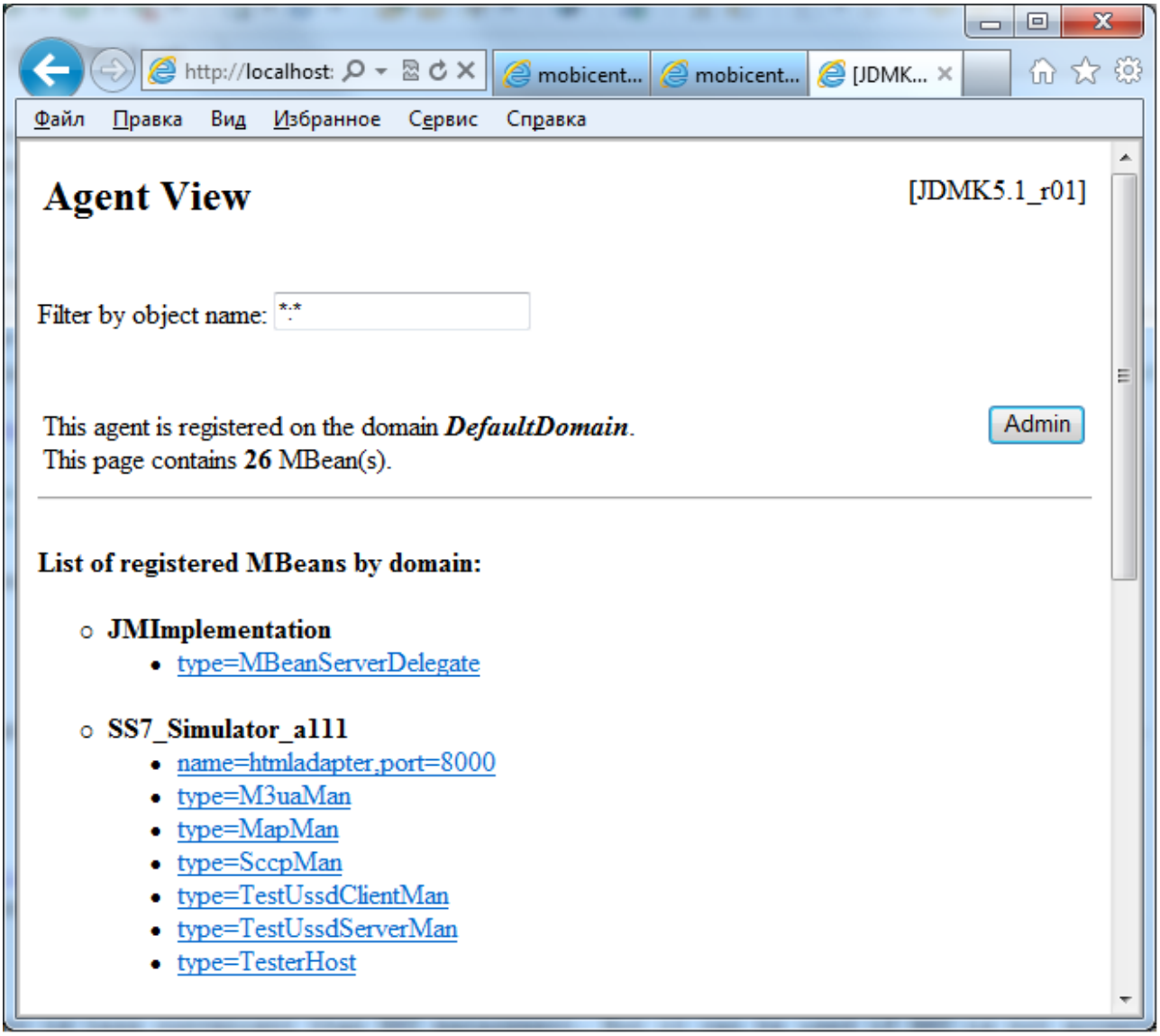


Figure 10. Managing remotely with a HTML Interface

2. The web page will display all Beans. "TesterHost" is the main MBean in which you can select the mode for test working, start/stop testing (buttons "Start"/"Stop") and quit a tester host (button "quit"). You can use other Beans for setting different options for layers and test tasks.

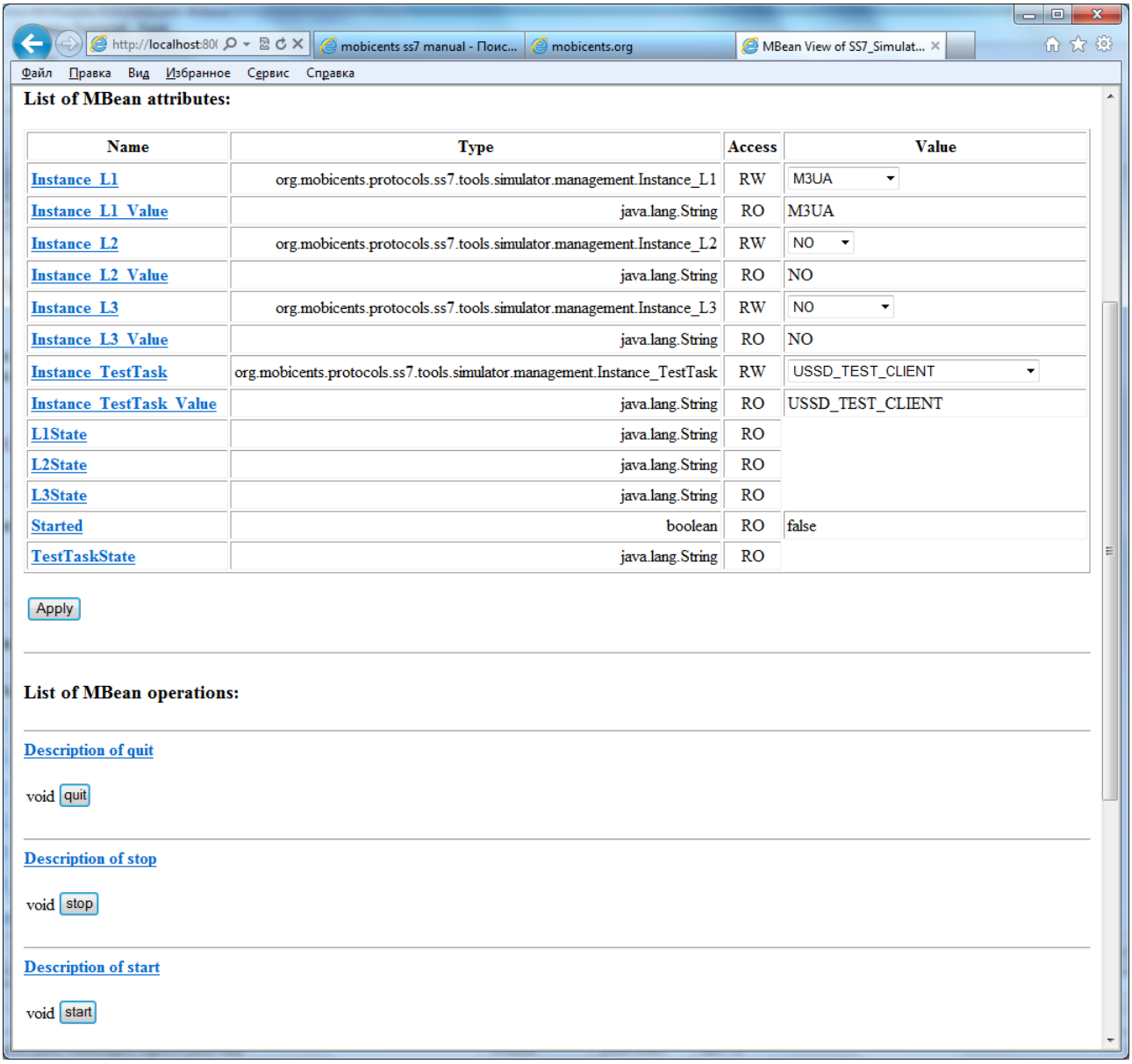


Figure 11. TesterHost Bean

3. The Results of the tests can be found at the console (in the server) or in the log file (file name is "a1.log" where "a1" is the name of the Simulator instance).

Chapter 5. Configuring jboss-beans.xml

5.1. Configuring Restcomm SS7 Service

Configuration is done through an XML descriptor file named *jboss-beans.xml* located at *$JBOSS_HOME/server/profile_name/deploy/restcomm-ss7-service/META-INF*, where `profile_name` is the name of the server profile. Default *jboss-beans.xml* contains only m3ua usage. Templates for usage of Dialogic boards or m3ua and Dialogic boards together can be found in the release binaries in folders: *ss7/template/META-INF-dialogic* and *ss7/template/META-INF-m3ua-dialogic*

Restcomm SS7 Layer 4 (`SCCP, ISUP`) leverages either of the following `MTP` layers to exchange signaling messages with remote signaling points:

- `M3UA`
- `dahdi`
- `dialogic`

5.1.1. Configuring M3UA

You must configure `M3UAManagement` if the underlying SS7 service will leverage `M3UA`. For more details on configuring `M3UAManagement`, please refer to M3UA Management.

Scroll down to the section for M3UA Layer in the *jboss-beans.xml* file and define the properties to suit your requirements.

```
        <!-- ===================================================================== -->
        <!-- SCTP Properties                                                       -->
        <!-- Used by M3UA layer                                                    -->
        <!-- ===================================================================== -->
        <bean name="SCTPManagement"
class="org.mobicents.protocols.sctp.netty.NettySctpManagementImpl">
                <constructor>
                        <parameter>SCTPManagement</parameter>
                </constructor>
                <property name="persistDir">${jboss.server.data.dir}</property>
        </bean>

        <bean name="SCTPShellExecutor"
                class="org.mobicents.protocols.ss7.m3ua.impl.oam.SCTPShellExecutor">
            <property name="sctpManagements">
                <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.sctp.netty.NettySctpManagementImpl">
                    <entry>
                        <key>SCTPManagement</key>
                        <value>
                            <inject bean="SCTPManagement" />
                        </value>
                    </entry>
```

```
                </map>
            </property>
        </bean>

        <!-- ===================================================================== -->
        <!-- M3UA -->
        <!-- M3UAManagement is managing the m3ua side commands -->
        <!-- ===================================================================== -->
        <!-- -->
        <bean name="Mtp3UserPart"
class="org.mobicents.protocols.ss7.m3ua.impl.M3UAManagementImpl">
            <constructor>
                <parameter>Mtp3UserPart</parameter>
                <parameter>{this-platform}-jSS7</parameter>
            </constructor>
            <property name="persistDir">${jboss.server.data.dir}</property>
            <property name="maxSequenceNumber">256</property>
            <property name="maxAsForRoute">2</property>
            <property name="deliveryMessageThreadCount">1</property>
            <property name="routingLabelFormat">
                <inject bean="RoutingLabelFormat" />
            </property>
            <property name="transportManagement">
                <inject bean="SCTPManagement" />
            </property>
        </bean>

        <bean name="M3UAShellExecutor"
                class="org.mobicents.protocols.ss7.m3ua.impl.oam.M3UAShellExecutor">
            <property name="m3uaManagements">
                <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.ss7.m3ua.impl.M3UAManagementImpl">
                    <entry>
                        <key>Mtp3UserPart</key>
                        <value>
                            <inject bean="Mtp3UserPart" />
                        </value>
                    </entry>
                </map>
            </property>
        </bean>
```

`org.mobicents.protocols.sctp.netty.NettySctpManagementImpl`

This SCTP Management Bean takes a `String` as a constructor argument. The name is prepended to the name of the XML file created by the SCTP stack for persisting the state of SCTP resources. This XML file is stored in the path specified by the property `persistDir`. For example, in the above case, when Restcomm SS7 Service is started, a file named *SCTPManagement_sctp.xml* will be created at *$JBOSS_HOME/server/profile_name/data* directory. The other properties of the Stack are defined below:

`org.mobicents.protocols.ss7.m3ua.impl.M3UAManagementImpl`

This M3UA Management Bean takes a `String` as a first constructor argument. The name is prepended to the name of the XML file created by the M3UA stack for persisting the state of M3UA resources. The second constructor argument of M3UA Management Bean is also a `String`. This is a productName parameter.

This XML file is stored in the path specified by the property `persistDir`. For example, in the above case, when Restcomm SS7 Service is started, a file named *Mtp3UserPart_m3ua1.xml* will be created at *$JBOSS_HOME/server/profile_name/data* directory. The other properties of the Stack are defined below:

persistDir

As explained above

routingLabelFormat

The routing label format supported by this network. See Configuring MTP3 routing label for further details.

transportManagement

SCTPManagement mbean should be provided here.

other parameters

See M3UA stack properties

5.1.2. Configuring dahdi

`Dahdi` based MTP layer will only be used if you have installed `dahdi` based SS7 hardware (Sangoma or Diguim) cards. `DahdiLinksetFactory` is responsible for creating new instances of `DahdiLinkset` when instructed by the `LinksetManager`.

The corresponding native libraries for `dahdi` (`libmobicents-dahdi-linux` library) must be compiled before stack configuring and put into `$JBOSS_HOME/bin/META-INF/lib/linux2/x86` if OS is 32 bit or `$JBOSS_HOME/bin/META-INF/lib/linux2/x64` if OS and JAVA is 64 bit.

See JSS7 Installation Guide for more details.

Libraries are compiled only for linux OS for 32-bit JAVA for now.

```
    <bean name="DahdiLinksetFactory"
class="org.mobicents.ss7.hardware.dahdi.oam.DahdiLinksetFactory">
    </bean>
```

`LinksetFactoryFactory` is just a call-back class listening for new factories deployed. It maintains a Map of available 'factory name' vs 'factory'. You should never touch this bean.

`LinksetManager` is responsible for managing `Linkset` and `Link`.

```
    <!-- ============================================== -->
    <!-- Linkset manager Service                        -->
    <!-- ============================================== -->
    <bean name="LinksetFactoryFactory"
class="org.mobicents.ss7.linkset.oam.LinksetFactoryFactory">
            <incallback method="addFactory" />
            <uncallback method="removeFactory" />
    </bean>

    <bean name="DahdiLinksetFactory"
class="org.mobicents.ss7.hardware.dahdi.oam.DahdiLinksetFactory">
    </bean>

    <bean name="LinksetManager"
class="org.mobicents.ss7.linkset.oam.LinksetManagerImpl">
        <constructor>
            <parameter>LinksetManager</parameter>
        </constructor>
        <property name="scheduler">
            <inject bean="SS7Scheduler" />
        </property>
        <property name="linksetFactoryFactory">
            <inject bean="LinksetFactoryFactory" />
        </property>
        <property name="persistDir">${jboss.server.data.dir}</property>
    </bean>

    <bean name="LinksetExecutor" class=
"org.mobicents.ss7.linkset.oam.LinksetExecutor">
        <property name="linksetManager">
            <inject bean="LinksetManager" />
        </property>
    </bean>
```

When LinksetManagerImpl is started it looks for the file *linksetmanager.xml* containing serialized information about underlying linksets and links. The directory path is configurable by changing the value of the property `persistDir`.

linksetmanager.xml should never be edited by you manually. Always use the Shell Client to connect to the Stack and execute appropriate commands.

`LinksetExecutor` accepts the `linkset` commands and executes necessary operations.

5.1.3. Configuring dialogic

`Dialogic` based MTP layer will only be used if you have installed Dialogic cards. `DialogicMtp3UserPart` communicates with Dialogic hardware. It is assumed here that MTP3 and MTP2 is leveraged from the Dialogic Stack either on-board or on-host.

The corresponding native libraries for `dialogic` (native lib `libgctjni.so` and gctApi library `gctApi.jar`) should be downloaded from the Dialogic site and copied : * `libgctjni.so` - to the folder `$JBOSS_HOME/bin/META-INF/lib/linux2/x86` if OS is 32 bit or `$JBOSS_HOME/bin/META-INF/lib/linux2/x64` if OS and JAVA is 64 bit. * `gctApi.jar` - to the folder `jboss-5.1.0.GA/server/default/deploy/Restcomm-ss7-service/lib`

See JSS7 Installation Guide for more details.

```
    <!-- ================================================================== -->
    <!-- Dialogic Mtp3UserPart -->
    <!-- ================================================================== -->
    <!---->
    <bean name="Mtp3UserPart"
class="org.mobicents.ss7.hardware.dialogic.DialogicMtp3UserPart">
        <constructor>
            <parameter>{this-platform}-jSS7</parameter>
        </constructor>
            <property name="sourceModuleId">61</property>
            <property name="destinationModuleId">34</property>
            <property name="routingLabelFormat">
                    <inject bean="RoutingLabelFormat" />
            </property>
    </bean>
```

This Dialogic Bean takes a `String` as a first constructor argument. This is a productName parameter.

The other properties of the Stack are defined below:

sourceModuleId

`sourceModuleId` is the id of source module and should match with the value configured in the file *system.txt* used by `dialogic` drivers. In the above example, 61 is assigned for mobicents process.

destinationModuleId

`destinationModuleId` is the id of destination module. In the above example, 34 is the id of Dialogic MTP3 module.

routingLabelFormat

The routing label format supported by this network. See Configuring MTP3 routing label for further details.

5.1.4. Configuring MTP3 routing label

MTP Level 3 routes messages based on the routing label in the signaling information field (SIF) of message signal units. The routing label is comprised of the destination point code (DPC), originating point code (OPC), and signaling link selection (SLS) field. Overtime different standards cameup with different routing label format. For example An ANSI routing label uses 7 octets; an ITU-T routing label uses 4 octets.

Restcomm jSS7 is flexible to configure the routing label as shown below.

```
        <!-- ================================================================= -->
        <!-- MTP3 Properties -->
        <!-- Define MTP3 routing label Format -->
        <!-- ================================================================= -->
        <bean name="RoutingLabelFormat"
class="org.mobicents.protocols.ss7.mtp.RoutingLabelFormat">
                <constructor
factoryClass="org.mobicents.protocols.ss7.mtp.RoutingLabelFormat"
                        factoryMethod="getInstance">
                        <parameter>ITU</parameter>
                </constructor>
        </bean>
```

Following table shows various routing formats supported

Table 3. Routing Format

Name	point code length	sls length
ITU	14-bits	4-bits
ANSI_Sls8Bit	24-bits	8-bits
ANSI_Sls5Bit	24-bits	5-bits
Japan_TTC_DDI	not supported yet	not supported yet
Japan_NTT	not supported yet	not supported yet
China	not supported yet	not supported yet

5.1.5. Configuring SCCP

As name suggests `SccpStack` initiates the SCCP stack routines.

```
    <!-- ================================================================== -->
    <!-- SCCP Service -->
    <!-- ================================================================== -->
    <bean name="SccpStack" class=
"org.mobicents.protocols.ss7.sccp.impl.SccpStackImpl">
        <constructor>
            <parameter>SccpStack</parameter>
        </constructor>
        <property name="persistDir">${jboss.server.data.dir}</property>
        <property name="mtp3UserParts">
            <map keyClass="java.lang.Integer"
valueClass="org.mobicents.protocols.ss7.mtp.Mtp3UserPart">
                <entry>
                    <key>1</key>
                        <value>
                                <inject bean="Mtp3UserPart" />
                        </value>
                </entry>
            </map>
        </property>
    </bean>

    <bean name="SccpExecutor"
class="org.mobicents.protocols.ss7.sccp.impl.oam.SccpExecutor">
        <property name="sccpStacks">
            <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.ss7.sccp.impl.SccpStackImpl">
                <entry>
                    <key>SccpStack</key>
                    <value>
                        <inject bean="SccpStack" />
                    </value>
                </entry>
            </map>
        </property>
    </bean>
```

`org.mobicents.protocols.ss7.sccp.impl.SccpStackImpl` takes `String` as constructor argument. The name is prepend to `xml` file created by SCCP stack for persisting state of SCCP resources. The `xml` is stored in path specified by `persistDir` property above.

For example in above case, when Restcomm SS7 Service is started 3 file's *SccpStack_management2.xml*, *SccpStack_sccpresource2.xml* and *SccpStack_sccprouter2.xml* will be created at *$JBOSS_HOME/server/profile_name/data* directory

Stack has following properties:

persistDir

As explained above

mtp3UserParts

specifies SS7 Level 3 to be used as transport medium(be it SS7 card or M3UA). Restcomm jSS7 SCCP allows configuring multiple MTP3 layers for same SCCP stack. This allows to have multiple local point-code and connecting to various networks while SCCP layer remains same

`SccpExecutor` accepts `sccp` commands and executes necessary operations

5.1.6. Configuring TCAP

`TcapStack` initiates the TCAP stack routines. Respective TCAP stack beans are instantiated for each `MAP`, `CAP` Service. If you are using either one, feel free to delete the other.

```
    <!-- ================================================================== -->
    <!-- TCAP Service -->
    <!-- ================================================================== -->
    <bean name="TcapStackMap" class="org.mobicents.protocols.ss7.tcap.TCAPStackImpl">
        <constructor>
            <parameter>TcapStackMap</parameter>
            <parameter>
                <inject bean="SccpStack" property="sccpProvider" />
            </parameter>
            <parameter>8</parameter>
        </constructor>
        <property name="persistDir">${jboss.server.data.dir}</property>
        <property name="previewMode">false</property>
    </bean>

    <bean name="TcapStackCap" class="org.mobicents.protocols.ss7.tcap.TCAPStackImpl">
        <constructor>
            <parameter>TcapStackCap</parameter>
                <parameter>
                    <inject bean="SccpStack" property="sccpProvider" />
                </parameter>
                <parameter>146</parameter>
            </constructor>
        <property name="persistDir">${jboss.server.data.dir}</property>
        <property name="previewMode">false</property>
    </bean>

    <bean name="TcapStack" class="org.mobicents.protocols.ss7.tcap.TCAPStackImpl">
        <constructor>
            <parameter>TcapStack</parameter>
            <parameter>
                <inject bean="SccpStack" property="sccpProvider" />
            </parameter>
            <parameter>9</parameter>
        </constructor>
        <property name="persistDir">${jboss.server.data.dir}</property>
        <property name="previewMode">false</property>
    </bean>
```

```
    <bean name="TcapExecutor"
class="org.mobicents.protocols.ss7.tcap.oam.TCAPExecutor">
        <property name="tcapStacks">
            <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.ss7.tcap.TCAPStackImpl">
                <entry>
                    <key>TcapStackMap</key>
                    <value>
                        <inject bean="TcapStackMap" />
                    </value>
                </entry>
                <entry>
                    <key>TcapStackCap</key>
                    <value>
                        <inject bean="TcapStackCap" />
                    </value>
                </entry>
                <entry>
                    <key>TcapStack</key>
                    <value>
                        <inject bean="TcapStack" />
                    </value>
                </entry>
            </map>
        </property>
    </bean>
```

`org.mobicents.protocols.ss7.tcap.TCAPStackImpl` takes `String` as a first constructor argument. The name is prepend to `xml` file created by TCAP stack for persisting state of TCAP resources. The `xml` is stored in path specified by `persistDir` property above.

For example in above case, when Restcomm SS7 Service is started 3 file's *TcapStack_management.xml*, *TcapStack_managementMap.xml* and *TcapStack_managementCap.xml* will be created at *$JBOSS_HOME/server/profile_name/data* directory. Then `org.mobicents.protocols.ss7.tcap.TCAPStackImpl` takes `SccpStack` as second constructor argument. TCAP uses passed SCCP stack. Constructor also takes the sub system number (SSN) which is registered with passed SCCP stack (this is the third parameter).

TCAP Stack has following configurable properties:

persistDir

As explained above

previewMode: public void setPreviewMode(boolean val);

PreviewMode is needed for special processing mode. By default TCAP is not set in PreviewMode. When PreviewMode set in TCAP level:

- Stack only listens for incoming messages and does not send anything. The methods `send()`, `close()`, `sendComponent()` and other such methods do nothing.

- A TCAP Dialog is temporary. TCAP Dialog is discarded after any incoming message like TC-BEGIN or TC-CONTINUE has been processed.
- For any incoming messages (including TC-CONTINUE, TC-END, TC-ABORT) a new TCAP Dialog is created (and then deleted).
- There are no timers and timeouts.

`TcapExecutor` accepts `tcap` commands and executes necessary operations

5.1.7. Configuring ShellExecutor

`ShellExecutor` is responsible for listening incoming commands. Received commands are executed on local resources to perform actions like creation and management of `TCAP`, `SCCP`, `SCTP` and `M3UA` stack.

```
    <!-- ================================================================== -->
    <!-- Shell Service -->
    <!-- ================================================================== -->
    <!-- Define Shell Executor -->
    <bean name="ShellExecutor"
class="com.mobicents.ss7.management.console.ShellServer">
        <constructor>
            <parameter>
                <inject bean="SS7Scheduler" />
            </parameter>
            <parameter>
                <list class="javolution.util.FastList"
elementClass="org.mobicents.ss7.management.console.ShellExecutor">
                    <inject bean="SccpExecutor" />
                    <inject bean="M3UAShellExecutor" />
                    <inject bean="SCTPShellExecutor" />
                    <inject bean="TcapExecutor" />
                    <!-- <inject bean="LinksetExecutor" /> -->
                </list>
            </parameter>
        </constructor>

        <property name="address">${jboss.bind.address}</property>
        <property name="port">3435</property>
        <property name="securityDomain">java:/jaas/jmx-console</property>
    </bean>
```

By default ShellExecutor listens at `jboss.bind.address` and port `3435`. (This is used when you use CLI access after running of ss7-cli command). You may set the `address` property to any valid IP address that your host is assigned. The shell commands are exchanged over TCP/IP.

To understand JBoss bind options look at Installation_And_Getting_Started_Guide

`SCTPShellExecutor` and `M3UAShellExecutor` is declared only if MTP layer `M3UA` is used. If `dialogic` MTP

layer is used these beans are not decalred and should be removed from FastList too. For dahdi need to declare LinksetExecutor bean and add in FastList above.

5.1.8. Configuring MAP

MapStack initiates the MAP stack routines.

```
    <!-- ================================================================= -->
    <!-- MAP Service -->
    <!-- ================================================================= -->
    <bean name="MapStack" class="org.mobicents.protocols.ss7.map.MAPStackImpl">
        <constructor>
            <parameter>MapStack</parameter>
            <parameter>
                <inject bean="TcapStackMap" property="provider" />
            </parameter>
        </constructor>
    </bean>
```

org.mobicents.protocols.ss7.tcap.MAPStackImpl takes String as a first constructor argument. The name is prepend to xml file created by MAP stack for persisting state of MAP resources. The xml is stored in path specified by persistDir property above. As for now MAP stack does not store any XML files. The second constructor argument is TcapStack. MAP uses passed TCAP stack.

Feel free to delete declaration of this bean if your service is not consuming MAP messages.

5.1.9. Configuring CAP

CapStack initiates the CAP stack routines.

```
    <!-- ================================================================= -->
    <!-- CAP Service -->
    <!-- ================================================================= -->
    <bean name="CapStack" class="org.mobicents.protocols.ss7.cap.CAPStackImpl">
        <constructor>
            <parameter>CapStack</parameter>
            <parameter>
                <inject bean="TcapStackCap" property="provider" />
            </parameter>
        </constructor>
    </bean>
```

org.mobicents.protocols.ss7.tcap.CAPStackImpl takes String as a first constructor argument. The name is prepend to xml file created by CAP stack for persisting state of CAP resources. The xml is stored in path specified by persistDir property above. As for now CAP stack does not store any XML files. The second constructor argument is TcapStack. CAP uses passed TCAP stack.

Feel free to delete declaration of this bean if your service is not consuming CAP messages.

5.1.10. Configuring ISUP

`IsupStack` initiates the ISUP stack routines.

```
    <!-- ================================================================= -->
    <!-- ISUP Service -->
    <!-- ================================================================= -->
    <bean name="CircuitManager"
class="org.mobicents.protocols.ss7.isup.impl.CircuitManagerImpl">
    </bean>

    <bean name="IsupStack" class=
"org.mobicents.protocols.ss7.isup.impl.ISUPStackImpl">
        <constructor>
            <parameter>
                <inject bean="SS7Scheduler" />
            </parameter>
            <parameter>22234</parameter>
            <parameter>2</parameter>
        </constructor>
        <property name="mtp3UserPart">
            <inject bean="Mtp3UserPart" />
        </property>
        <property name="circuitManager">
            <inject bean="CircuitManager" />
        </property>
    </bean>
```

`org.mobicents.protocols.ss7.isup.impl.ISUPStackImpl` takes `SS7Scheduler`, local signaling pointcode and network indicator as constructor argument.

Stack has following properties:

mtp3UserPart
: specifies SS7 Level 3 to be used as transport medium (be it SS7 card or M3UA).

circuitManager
: CIC management bean

Feel free to delete declaration of this bean if your service is not consuming ISUP messages.

5.1.11. Configuring SS7Service

SS7Service acts as core engine binding all the components together.

```
    <!-- ================================================================= -->
    <!-- RestComm SS7 Service -->
    <!-- ================================================================= -->
    <bean name="TCAPSS7Service" class="org.mobicents.ss7.SS7Service">
```

```
        <constructor><parameter>TCAP</parameter></constructor>

<annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="org.mobicents.ss7:serv
ice=TCAPSS7Service",exposedInterface=org.mobicents.ss7.SS7ServiceMBean.class,registerD
irectly=true)
        </annotation>
        <property name="jndiName">java:/restcomm/ss7/tcap</property>
        <property name="stack">
            <inject bean="TcapStack" property="provider" />
        </property>
    </bean>
    <bean name="MAPSS7Service" class="org.mobicents.ss7.SS7Service">
        <constructor><parameter>MAP</parameter></constructor>

<annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="org.mobicents.ss7:serv
ice=MAPSS7Service",exposedInterface=org.mobicents.ss7.SS7ServiceMBean.class,registerDi
rectly=true)
        </annotation>
        <property name="jndiName">java:/restcomm/ss7/map</property>
        <property name="stack">
            <inject bean="MapStack" property="MAPProvider" />
        </property>
    </bean>
    <bean name="CAPSS7Service" class="org.mobicents.ss7.SS7Service">
        <constructor><parameter>CAP</parameter></constructor>

<annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="org.mobicents.ss7:serv
ice=CAPSS7Service",exposedInterface=org.mobicents.ss7.SS7ServiceMBean.class,registerDi
rectly=true)
        </annotation>
        <property name="jndiName">java:/restcomm/ss7/cap</property>
        <property name="stack">
            <inject bean="CapStack" property="CAPProvider" />
        </property>
    </bean>
    <bean name="ISUPSS7Service" class="org.mobicents.ss7.SS7Service">
        <constructor><parameter>ISUP</parameter></constructor>

<annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="org.mobicents.ss7:serv
ice=ISUPSS7Service",exposedInterface=org.mobicents.ss7.SS7ServiceMBean.class,registerD
irectly=true)
        </annotation>
        <property name="jndiName">java:/restcomm/ss7/isup</property>
        <property name="stack">
            <inject bean="IsupStack" property="isupProvider" />
        </property>
    </bean>
```

TCAPSS7Service binds TcapStack to JNDI `java:/restcomm/ss7/tcap`.

MAPSS7Service binds MapStack to JNDI `java:/restcomm/ss7/map`.

CAPSS7Service binds CapStack to JNDI `java:/restcomm/ss7/cap`.

ISUPSS7Service binds IsupStack to JNDI `java:/restcomm/ss7/isup`.

The JNDI name can be configured to any valid JNDI name specific to your application.

Feel free to delete service that your application is not using.

5.1.12. Configuring jSS7 Management Service

jSS7 Managemenet Service provides some extra functionality for stack management including jmx access to stacks, performance (statistics) and alarm management.

```
    <bean name="Ss7Management"
        class="org.mobicents.protocols.ss7.oam.common.jmxss7.Ss7Management">

<annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name="org.mobicents.ss7:serv
ice=Ss7Management",exposedInterface=org.mobicents.protocols.ss7.oam.common.jmxss7.Ss7M
anagementMBean.class,registerDirectly=true)</annotation>
        <property name="agentId">jboss</property>
    </bean>

    <bean name="RestcommAlarmManagement"
        class="org.mobicents.protocols.ss7.oam.common.alarm.AlarmProvider">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommStatisticManagement"

class="org.mobicents.protocols.ss7.oam.common.statistics.CounterProviderManagement">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
        </constructor>
        <property name="persistDir">${jboss.server.data.dir}</property>
    </bean>

    <bean name="RestcommSctpManagement"
        class="org.mobicents.protocols.ss7.oam.common.sctp.SctpManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
```

```
            </parameter>
            <parameter>
                <inject bean="SCTPManagement" />
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommM3uaManagement"
        class="org.mobicents.protocols.ss7.oam.common.m3ua.M3uaManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="Mtp3UserPart" />
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommSccpManagement"
        class="org.mobicents.protocols.ss7.oam.common.sccp.SccpManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="SccpStack" />
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommTcapManagement"
        class="org.mobicents.protocols.ss7.oam.common.tcap.TcapManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="TcapStack" />
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommTcapMapManagement"
        class="org.mobicents.protocols.ss7.oam.common.tcap.TcapManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="TcapStackMap" />
```

```
            </parameter>
        </constructor>
    </bean>

    <bean name="RestcommTcapCapManagement"
        class="org.mobicents.protocols.ss7.oam.common.tcap.TcapManagementJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="TcapStackCap" />
            </parameter>
        </constructor>
    </bean>

<!--
    <bean name="RestcommLinksetManagement"
        class="org.mobicents.protocols.ss7.oam.common.linkset.LinksetManagerJmx">
        <constructor>
            <parameter>
                <inject bean="Ss7Management" />
            </parameter>
            <parameter>
                <inject bean="LinksetManager" />
            </parameter>
        </constructor>
    </bean>
-->
```

5.2. Configuring Restcomm Signaling Gateway

Configuration is done through an XML descriptor named *sgw-beans.xml* and is located at *restcomm-ss7-sgw/deploy*

Before Restcomm Signaling Gateway is configured the corresponding native libraries for `dahdi` or `dialogic` should be copied to `restcomm-ss7-sgw/native/32` or `restcomm-ss7-sgw/native/64` folders and gctApi library to `restcomm-ss7-sgw/lib` folder. See more details for where to get native libraries in Configuring dahdi and Configuring dialogic.

5.2.1. Configuring M3UA (Signaling Gateway)

SGW will expose the SS7 signals received from legacy network to IP network over M3UA

```
    <bean name="SCTPManagement" class="org.mobicents.protocols.sctp.ManagementImpl">
        <constructor>
            <parameter>SCTPManagement</parameter>
        </constructor>
        <property name="persistDir">${sgw.home.dir}/ss7</property>
    </bean>

    <bean name="SCTPShellExecutor"
class="org.mobicents.protocols.ss7.m3ua.impl.oam.SCTPShellExecutor">
        <property name="sctpManagements">
            <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.sctp.ManagementImpl">
                <entry>
                    <key>SCTPManagement</key>
                    <value>
                        <inject bean="SCTPManagement" />
                    </value>
                </entry>
            </map>
        </property>
    </bean>

    <bean name="Mtp3UserPart"
class="org.mobicents.protocols.ss7.m3ua.impl.M3UAManagementImpl">
        <constructor>
            <parameter>Mtp3UserPart</parameter>
            <parameter>Restcomm-jSS7</parameter>
        </constructor>
        <property name="persistDir">${sgw.home.dir}/ss7</property>
        <property name="transportManagement">
            <inject bean="SCTPManagement" />
        </property>
    </bean>

    <bean name="M3UAShellExecutor"
class="org.mobicents.protocols.ss7.m3ua.impl.oam.M3UAShellExecutor">
        <property name="m3uaManagements">
            <map keyClass="java.lang.String"
valueClass="org.mobicents.protocols.ss7.m3ua.impl.M3UAManagementImpl">
                <entry>
                    <key>Mtp3UserPart</key>
                    <value>
                        <inject bean="Mtp3UserPart" />
                    </value>
                </entry>
            </map>
        </property>
    </bean>
```

5.2.2. Configuring LinksetFactory

Concrete implementation of `LinksetFactory` is responsible to create new instances of corresponding `Linkset` when instructed by `LinksetManager`. Restcomm Signaling Gateway defines two linkset factories :

- `DahdiLinksetFactory`

```
    <bean name="DahdiLinksetFactory"
        class="org.mobicents.ss7.hardware.dahdi.oam.DahdiLinksetFactory">
        <property name="scheduler">
            <inject bean="Scheduler" />
        </property>
    </bean>
```

- `DialogicLinksetFactory`

```
    <bean name="DialogicLinksetFactory"
        class="org.mobicents.ss7.hardware.dialogic.oam.DialogicLinksetFactory">
    </bean>
```

Its highly unlikely that you would require both the factories on same gateway. If you have `dahdi` based SS7 card installed, keep `DahdiLinksetFactory` and remove other. If you have `dialogic` based SS7 card installed, keep `DialogicLinksetFactory` and remove other.

`LinksetFactoryFactory` is just a call-back class listening for new factories deployed and maintains Map of available factory name vs factory. You should never touch this bean.

5.2.3. Configuring LinksetManager

`LinksetManager` is responsible for managing `Linkset` and `Link`.

```
    <!-- ============================================= -->
    <!-- Linkset manager Service                                    -->
    <!-- ============================================= -->
    <bean name="LinksetManager"
class="org.mobicents.ss7.linkset.oam.LinksetManagerImpl">
        <constructor>
            <parameter>LinksetManager</parameter>
        </constructor>
        <property name="scheduler">
            <inject bean="Scheduler" />
        </property>

        <property name="linksetFactoryFactory">
            <inject bean="LinksetFactoryFactory" />
        </property>
        <property name="persistDir">${sgw.home.dir}/ss7</property>
    </bean>

    <bean name="LinksetExecutor" class=
"org.mobicents.ss7.linkset.oam.LinksetExecutor">
        <property name="linksetManager">
            <inject bean="LinksetManager" />
        </property>
    </bean>
```

LinksetManagerImpl when started looks for file *linksetmanager.xml* containing serialized information about underlying linksets and links. The directory path is configurable by changing value of `persistDir` property.

linksetmanager.xml should never be edited by hand. Always use Shell Client to connect to Restcomm Signaling Gateway and execute commands.

`LinksetExecutor` accepts the `linkset` commands and executes necessary operations.

5.2.4. Configuring ShellExecutor

`ShellExecutor` is responsible for listening to incoming command. Received commands are executed on local resources to perform actions like creation and management of `Linkset`, management of `M3UA` stack.

```
    <!-- ============================================= -->
    <!-- Shell Service                                 -->
    <!-- ============================================= -->
    <bean name="ShellExecutor"
class="org.mobicents.ss7.management.console.ShellServer">
        <constructor>
            <parameter>
                <inject bean="Scheduler" />
            </parameter>
            <parameter>
                <list class="javolution.util.FastList"
elementClass="org.mobicents.ss7.management.console.ShellExecutor">
                    <inject bean="M3UAShellExecutor" />
                    <inject bean="SCTPShellExecutor" />
                    <inject bean="LinksetExecutor" />
                </list>
            </parameter>
        </constructor>

        <property name="address">${sgw.bind.address}</property>
        <property name="port">3435</property>
    </bean>
```

By default ShellExecutor listens at `sgw.bind.address` and port `3435`. You may set the `address` property to any valid IP address that your host is assigned. The shell commands are exchanged over TCP/IP.

5.2.5. Configuring SignalingGateway

`SignalingGateway` acts as core engine binding all the components together.

```
    <!-- ============================================= -->
    <!-- mobicents Signaling Gateway                   -->
    <!-- ============================================= -->
    <bean name="SignalingGateway" class="org.mobicents.ss7.sgw.SignalingGateway">
        <property name="scheduler">
            <inject bean="Scheduler" />
        </property>

        <property name="nodalInterworkingFunction">
            <inject bean="NodalInterworkingFunction" />
        </property>

    </bean>
```

The `NodalInterworkingFunction` sits between the SS7 netwrok and IP network and routes messages to/from both the MTP3 and the M3UA layer, based on the SS7 DPC or DPC/SI address information

Chapter 6. Configuring as a Standalone library

If you intend to use the Stack as a standalone library without using JBoss Application Server or JSLEE RAs, then you must manually build each of the protocols, configure them individually, and bind them together.

6.1. Building SCTP

This is an example of a typical SCTP startup procedure, in one line of code, using automatic configuration file detection:

```
    org.mobicents.protocols.api.Management sctpManagement = null;
    sctpManagement = new org.mobicents.protocols.sctp.ManagementImpl("Client");
    sctpManagement.setPersistDir("<your directory path>");

    sctpManagement.start();

    sctpManagement.setConnectDelay(10000);
```

How did jSS7 know where the configuration file was located and which one to load?

When `sctpManagement.start()` is called, jSS7 searches for a file named *Client_sctp.xml* in the directory path set by user by calling `sctpManagement.setPersistDir("<your directory path>")`. For example in case of linux you can pass something like `this.sctpManagement.setPersistDir("/home/abhayani/workarea/mobicents/git/jss7/master/map/load/client")`. If directory path is not set, Management searches for system property `sctp.persist.dir` to get the path for directory. Even if `sctp.persist.dir` system property is not set, Management will look at System set property `user.dir`.

Once you know SCTP layer is configured and started, you can set Stack level parameters like `connectionDelay` by calling setters like: `sctpManagement.setConnectDelay(10000);`

Next step is adding of the Association and/or Server depending on whether this setup will be acting as client or server or both. Normally a configured set of Association and/or Server are stored in the xml config file and this way is recommended. But if you want to remove all previously configured Associations and Servers you need to call this command (after a Stack start).

```
    sctpManagement.removeAllResourses();
```

Before adding server side association the server should also be defined and started as below:

```
    boolean acceptAnonymousConnections = false;
    int maxConcurrentConnectionsCount = 0; // for AnonymousConnections, 0 means
unlimited
    String[] extraHostAddresses = null; // for multi-homing
    IpChannelType ipChannelType = IpChannelType.SCTP;

    sctpManagement.addServer(SERVER_NAME, SERVER_IP, SERVER_PORT, ipChannelType,
acceptAnonymousConnections, maxConcurrentConnectionsCount, extraHostAddresses);
    sctpManagement.addServerAssociation(PEER_IP, PEER_PORT, SERVER_NAME,
SERVER_ASSOCIATION_NAME, ipChannelType);
    sctpManagement.startServer(SERVER_NAME);
```

For Client side association the code is the following:

```
    String[] extraHostAddresses = null; // for multi-homing
    IpChannelType ipChannelType = IpChannelType.SCTP;

    sctpManagement.addAssociation(HOST_IP, HOST_PORT, PEER_IP, PEER_PORT,
CLIENT_ASSOCIATION_NAME, ipChannelType, extraHostAddresses);
```

You should never start the Association programatically. Association will be started automatically when layer above M3UA's ASP is started.

This completes the SCTP configuration and start-up.

6.2. Building M3UA

Configuring the M3UA layer is similar to the steps followed for SCTP.

```
    org.mobicents.protocols.ss7.m3ua.impl clientM3UAMgmt = null;
    clientM3UAMgmt = new M3UAManagement("Client", null);
    clientM3UAMgmt.setPersistDir("<your directory path>");
    clientM3UAMgmt.setTransportManagement(sctpManagement);

clientM3UAMgmt.setDeliveryMessageThreadCount(DELIVERY_TRANSFER_MESSAGE_THREAD_COUNT);

    clientM3UAMgmt.start();
```

For M3UA, it should know which underlying SCTP layer to use `clientM3UAMgmt.setTransportManagement(sctpManagement);`.

Once M3UA is configured and started, next step is to add the As, Asp and routing rules for M3UA. Configured set of As, Asp and routing rules is trored in *Client_m3ua1.xml* (in "your directory path" folder). But if you want to remove all previously configured As, Asp and routing rules you need to call this command (after a Stack start).

```
    clientM3UAMgmt.removeAllResourses();
```

A set of As, Asp and routing rules depends on whether stack acts as Application Server side or Signaling Gateway side or just peer-to-peer (IPSP) client/server side. Below is an example of IPSP peer acting as client.

```
    ParameterFactoryImpl factory = new ParameterFactoryImpl();

    // Step 1 : Create App Server
    RoutingContext rc = factory.createRoutingContext(new long[] { 100l });
    TrafficModeType trafficModeType =
factory.createTrafficModeType(TrafficModeType.Loadshare);
    As as = clientM3UAMgmt.createAs("AS1", Functionality.AS, ExchangeType.SE,
IPSPType.CLIENT, rc, trafficModeType, 1, null);

    // Step 2 : Create ASP
    AspFactory aspFactor = clientM3UAMgmt.createAspFactory("ASP1",
CLIENT_ASSOCIATION_NAME);

    // Step3 : Assign ASP to AS
    Asp asp = clientM3UAMgmt.assignAspToAs("AS1", "ASP1");

    // Step 4: Add Route. Remote point code is 2
    clientM3UAMgmt.addRoute(SERVET_SPC, -1, -1, "AS1");
```

This completes the M3UA configuration and start-up. Once M3UA is configured depending on whether you are trying to build voice application that depends on ISUP or advanced network features such as those offered by supplementary services that depends on MAP, you would configure ISUP or SCCP

6.3. Building SCCP

Configuring the SCCP layer follows exactly same architecture of persisting configuration in xml file.

```
    org.mobicents.protocols.ss7.sccp.SccpStack sccpStack = null;
    sccpStack = new SccpStackImpl("MapLoadClientSccpStack");
    clientM3UAMgmt.setPersistDir("<your directory path>");
    sccpStack.setMtp3UserPart(1, this.clientM3UAMgmt);

    sccpStack.start();
```

Before starting SCCP stack all it needs to know is underlying MTP3 layer. Above sections explained building SCTP and M3UA, however if you are using Dialogic boards or dahdi based boards (Diguim/Sangoma), you need to build and configure respective MTP3 layers depending on hardware used and set those in SCCP Stack `sccpStack.setMtp3UserPart(1, this.clientM3UAMgmt)`.

One of the best features of jSS7 is it supports multiple MTP3 layers and hence you can have combination of many MTP3 layers (each of different or same type like M3UA, Dialogic and Dahid; all used at same time).

Once SCCP stack is started, it should be configured for local and remote signaling point-code, network indicator, remote sub system number and routing rules. For removing all previously configured resources you need to call this command (after a Stack start).

```
    sccpStack.removeAllResourses();
```

Here is an example for adding resources:

```
    sccpStack.getSccpResource().addRemoteSpc(0, SERVET_SPC, 0, 0);
    sccpStack.getSccpResource().addRemoteSsn(0, SERVET_SPC, SSN, 0, false);

    sccpStack.getRouter().addMtp3ServiceAccessPoint(1, 1, CLIENT_SPC,
NETWORK_INDICATOR, 0);
    sccpStack.getRouter().addMtp3Destination(1, 1, SERVET_SPC, SERVET_SPC, 0, 255,
255);

    ParameterFactoryImpl fact = new ParameterFactoryImpl();
    EncodingScheme ec = new BCDEvenEncodingScheme();
    GlobalTitle gt1 = fact.createGlobalTitle("-", 0,
org.mobicents.protocols.ss7.indicator.NumberingPlan.ISDN_TELEPHONY, ec,
NatureOfAddress.INTERNATIONAL);
    GlobalTitle gt2 = fact.createGlobalTitle("-", 0,
org.mobicents.protocols.ss7.indicator.NumberingPlan.ISDN_TELEPHONY, ec,
NatureOfAddress.INTERNATIONAL);
    SccpAddress localAddress = new
SccpAddressImpl(RoutingIndicator.ROUTING_BASED_ON_GLOBAL_TITLE, gt1, CLIENT_SPC, 0);
    sccpStack.getRouter().addRoutingAddress(1, localAddress);
    SccpAddress remoteAddress = new
SccpAddressImpl(RoutingIndicator.ROUTING_BASED_ON_GLOBAL_TITLE, gt2, SERVET_SPC, 0);
    sccpStack.getRouter().addRoutingAddress(2, remoteAddress);

    GlobalTitle gt = fact.createGlobalTitle("*", 0,
org.mobicents.protocols.ss7.indicator.NumberingPlan.ISDN_TELEPHONY, ec,
NatureOfAddress.INTERNATIONAL);
    SccpAddress pattern = new
SccpAddressImpl(RoutingIndicator.ROUTING_BASED_ON_GLOBAL_TITLE, gt, 0, 0);
    sccpStack.getRouter().addRule(1, RuleType.SOLITARY, LoadSharingAlgorithm.Bit0,
OriginationType.REMOTE, pattern, "K", 1, -1, null, 0);
    sccpStack.getRouter().addRule(2, RuleType.SOLITARY, LoadSharingAlgorithm.Bit0,
OriginationType.LOCAL, pattern, "K", 2, -1, null, 0);
```

Once SCCP is configured and started, next step it to build TCAP layer.

6.4. Building TCAP

There is no configuration to persist in case of TCAP.

```
    org.mobicents.protocols.ss7.tcap.api tcapStack = null;
    tcapStack = new TCAPStackImpl("Client", this.sccpStack.getSccpProvider(), SSN);

    tcapStack.start();

    this.tcapStack.setDialogIdleTimeout(60000);
    this.tcapStack.setMaxDialogs(MAX_DIALOGS);
```

Configuring TCAP is probably very simple as config reamins same irrespective of whether its used on client side or server side.

6.5. Building MAP

There is no configuration to persist in case of MAP; however MAP stack can take TCAPProvider from TCAPStack which is already configured for specific SSN as shown below:

```
    mapStack = new MAPStackImpl("Client", tcapStack.getProvider());
```

Or it can also directly take SccpProvider and pass SSN to MAP Stack as shown below. In this case MAPStack itself creates the TCAPStack and leverages TCAPProvider:

```
    mapStack = new MAPStackImpl("Client", sccpStack.getSccpProvider(), SSN);
```

Before MAPStack can be started, the Application interested in particualr MAP Service should register it-self as listener and activate that service:

```
    mapProvider = mapStack.getMAPProvider();
    mapProvider.addMAPDialogListener(this);
    mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
    mapProvider.getMAPServiceSupplementary().acivate();
    mapStack.start();
```

Below is how the Application code looks like:

```
    public class Client extends MAPDialogListener, MAPServiceSupplementaryListener  {
        //Implemet all MAPDialogListener methods here

        //Implement all MAPServiceSupplementaryListener methods here
    }
```

6.6. Common Code

All above snippet of code refers to below defined constants:

```
    // MTP Details
    protected final int CLIENT_SPC = 1;
    protected final int SERVET_SPC = 2;
    protected final int NETWORK_INDICATOR = 2;
    protected final int SERVICE_INIDCATOR = 3; //SCCP
    protected final int SSN = 8;

    protected final String CLIENT_IP = "127.0.0.1";
    protected final int CLIENT_PORT = 2345;

    protected final String SERVER_IP = "127.0.0.1";
    protected final int SERVER_PORT = 3434;

    protected final int ROUTING_CONTEXT = 100;

    protected final String SERVER_ASSOCIATION_NAME = "serverAsscoiation";
    protected final String CLIENT_ASSOCIATION_NAME = "clientAsscoiation";

    protected final String SERVER_NAME = "testserver";

    .....
    .....
```

Once you have completed development of your application, next thing is setting the classpath, compiling and starting application. You must set the classpath to point to restcomm-jss7-X.Y.Z/ss7/restcomm-ss7-service/lib. It has all the libraries needed to compile and start your application. Don’t forget to include your compiled Application class file in classpath before starting the Application.

Chapter 7. Managing Restcomm jSS7

Restcomm jSS7 comes with a convenient user-friendly Graphical User Interface (GUI) and a Command Line Interface (CLI) that will allow you to configure, monitor and manage the Stack. While the CLI tool allows complete configuration and control of the Stack, the GUI-based management enhances the usability of the platform and gives you the ability to create different SS7 configurations and manage the platform dynamically. This chapter will explain how to manage the Stack effectively using both the GUI and the CLI.

7.1. Linkset Management

7.1.1. Using CLI

You can manage Linksets and Links using CLI or GUI. You can create, delete, activate and deactivate linksets and links using the Shell command `linkset` with appropriate parameters. The `linkset` command can be used only when dahdi based cards are configured.

7.1.2. Using GUI

The GUI will allow you to manage your linksets and links efficiently using a user-friendly interface. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'linkset' link in the left panel. The main panel will display the names of all configured Linkset Management units. To configure or view the settings of a particular Linkset Management Unit you must click on the name of that unit. The GUI will look similar to the figure below and is divided into two tabs.

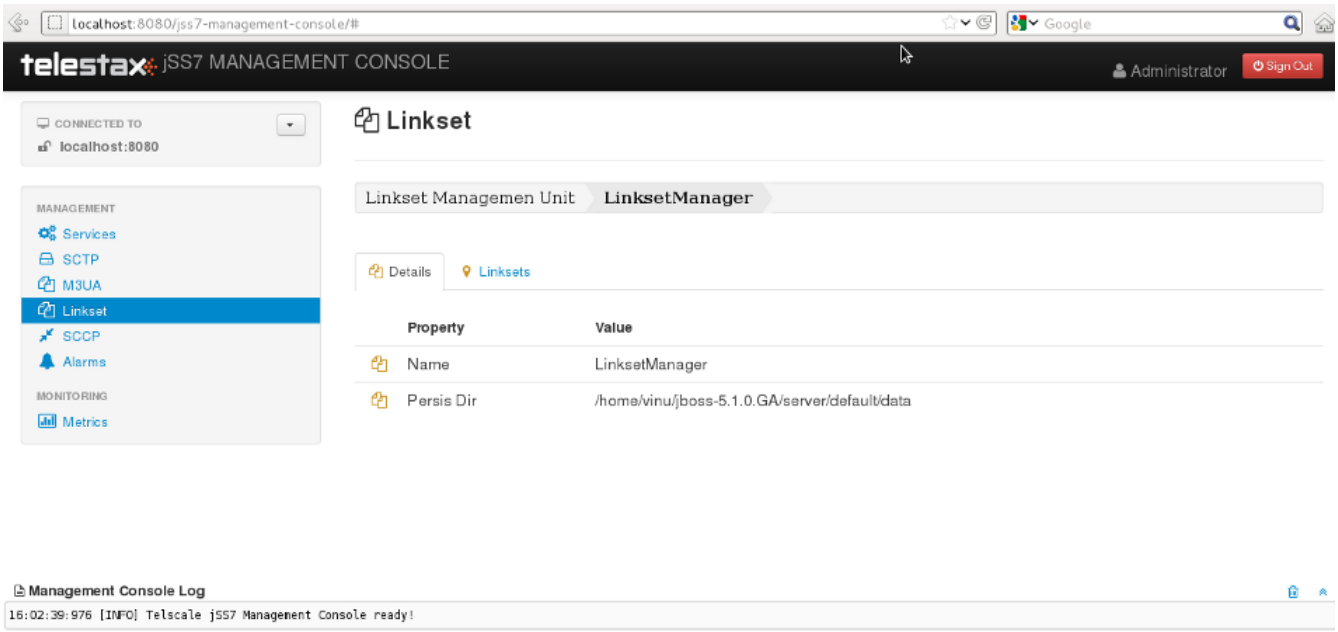


Figure 12. GUI - Linkset Management

The first tab will display the properties of the Linkset Management unit. These details displayed here are fetched from the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name. These properties cannot be modified here in the GUI. To modify them you

must modifiy the *jboss-beans.xml* and restart the Server. The GUI will then display the modified values.

The other tab will allow you to manage all Linksets and Links within this Linkset Management unit.

7.1.3. View all Linksets and Links

Using CLI

You can view the current status of all correctly configured Linksets and Links by issuing the command `linkset show` as described below:

```
Name
    linkset show

SYNOPSIS
    linkset show

DESCRIPTION
    This command is used to view all Links and Linksets and their states.

    The possible states of a Linkset are:

    - UNAVAILABLE   :   Indicates that the Linkset does not have any 'available'
                links and cannot transport traffic.

    - SHUTDOWN  :   Indicates that Linkset has been shutdown.

    - AVAILABLE :   Indicates that the Linkset has atleast one 'available'
                link and can transport traffic.

    The possible states of a Link are:

    - UNAVAILABLE   :   Indicates that the Link is not 'available' to transport
                traffic. A Link is 'unavailable' if the Link is remotely
                or locally inhibited by a user. It can also be
                'unavailable' if MTP2 has not been able to successfully
                activate the link connection.

    - SHUTDOWN  :   Indicates the Link has been shutdown in the configuration.

    - AVAILABLE :   Indicates the Link is active and 'available' to transport
                traffic.

    - FAILED    :   Indicates that the Link is not 'shutdown' but is
                'unavailable' at Layer 2 for some reason. For example, the
                Initial Alignment may fail, or the link test messages sent
                by MTP 3 are not being acknowledged.
```

Using GUI

Navigate to the specific Linkset Management unit and switch to the 'Linksets' tab. Here you can view a list of all the Linksets created. Every correctly configured Linkset will be displayed in a row and for each Linkset, the first column will display the name of the Linkset. The icon adjacent to the name will be lit 'green' if the Linkset is currently active or 'orange' if inactive. The second column will indicate the current mode of the Linkset (Configured / Not Configured), the third column will allow you to activate / deactivate the Linkset and the fourth column will allow you to delete the Linkset.

To view the details of all the Links created within a specific Linkset click on the name of the Linkset whose details you wish to view. This will launch the 'Links' view and display all the configured properties of the selected Linkset in the first tab. The second tab in this view will allow you to view all Links in this particular Linkset. You can click on any Link name here to view the configured properties. You can click on the bread crumbs at the top to return to any of the previous pages you navigated through.

7.1.4. Create a new Linkset

Using CLI

You can create a new Linkset by issuing the command `linkset create` with appropriate parameters as described below:

```
Name
    linkset create

SYNOPSIS
    linkset create dahdi opc <point-code> apc <point-code> ni <network-id>
    <linkset-name>

DESCRIPTION
    This command is used to create a new Linkset of type Dahdi as explained below. You
    must ensure that appropriate linkset factories are deployed prior to creating any
    linkset.

PARAMETERS
    opc <point-code>      -   MTP Point code of the originating
                        signalling point. Takes an Integer Value.

    apc <point-code>      -   MTP  Point code of the adjacent signalling
                        point. Takes an Integer Value.

    ni <network-id>       -   Network Identifier and should be one of
                        the following values
                            0 - International Network
                            1 - Spare (for International use)
                            2 - National Network
                            3 - Reserved for National use

    <linkset-name>            -   Name of the Linkset to be created. This
                        will be used to identify the linkset.
                        Takes a String value.

EXAMPLES
    linkset create dahdi opc 3 apc 4 ni 0 dahdilinkset1

    The above command will create a new linkset of type Dahdi and name dahdilinkset1.
    The originating point code value is 3 and the adjacent point code value is 4 and
    the network is an international network.
```

Using GUI

Procedure: Create new Linkset using GUI

1. In the section for Linksets in the Linkset Management Unit window, click on the 'Create Linkset' button. This will launch a pop-up 'Create Linkset'.
2. In the 'Create Linkset' pop-up, add details of the new Linkset. You must ensure that you fill in all the mandatory parameters (OPC, DPC, NI and Linkset Name). For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new Linkset will be created with parameters as specified. If there is an error in creating the Linkset then you will find the details of the error in the Management Console Log section below.

4. Click on the 'Close' button to close the 'Create Linkset' pop-up.

7.1.5. Remove a Linkset

Using CLI

You can delete an existing Linkset by issuing the command `linkset delete` with appropriate parameters as described below:

```
Name
    linkset delete

SYNOPSIS
    linkset delete <linkset-name>

DESCRIPTION
    This command is used to delete an existing Linkset.

PARAMETERS
    <linkset-name>        -   Name of the Linkset to be deleted.

EXAMPLE
    linkset delete dahdilinkset1

    The above command will delete the Linkset identified by the name dahdilinkset1.
```

Using GUI

Procedure: Delete Linkset using GUI

1. Navigate to the 'Linksets' section in the Linkset Management Unit window and locate the row corresponding to the Linkset you wish to delete.
2. You must ensure that the Linkset is deactivated prior to deletion. If the Linkset is deactivated, the last column for 'Delete' will display a 'x' button in red and will be enabled. If the Linkset is currently active, the 'x' button will be disabled. You can only delete the Linkset if it is not active.
3. Click on the red 'x' button to delete the corresponding Linkset.

7.1.6. Activate Linkset

Using CLI

You can activate an existing Linkset by issuing the command `linkset activate` with appropriate parameters as described below:

```
Name
    linkset activate

SYNOPSIS
    linkset activate <linkset-name>

DESCRIPTION
    This command is used to activate an existing Linkset.

PARAMETERS
    <linkset-name>        -   Name of the Linkset to be activated.

EXAMPLE
    linkset activate dahdilinkset1

    The above command will activate the Linkset identified by the name dahdilinkset1.
```

Using GUI

Procedure: Activate a Linkset using GUI

1. Navigate to the 'Linksets' section in the Linkset Management Unit window and locate the row corresponding to the Linkset you wish to activate.
2. Click on the 'Activate' button in the actions column to activate the corresponding Linkset.
3. If the Linkset has been activated successfully you will find the status indicating the Linkset as 'Available' and the Linkset's icon will be lit green. If there is an error and the Linkset failed to activate, you will find details of the error in the Management Console log below.

7.1.7. Deactivate Linkset

Using CLI

You can deactivate a currently active Linkset by issuing the command `linkset deactivate` with appropriate parameters as described below:

```
Name
    linkset deactivate

SYNOPSIS
    linkset deactivate <linkset-name>

DESCRIPTION
    This command is used to deactivate an existing Linkset.

PARAMETERS
    <linkset-name>        -   Name of the Linkset to be deactivated.

EXAMPLE
    linkset deactivate dahdilinkset1

    The above command will deactivate the Linkset identified by the name
dahdilinkset1.
```

Using GUI

Procedure: Deactivate a Linkset using GUI

1. Navigate to the 'Linksets' section in the Linkset Management Unit window and locate the row corresponding to the Linkset you wish to deactivate.
2. To deactivate a Linkset currently active, click on the 'Deactivate' button in the actions column of the row corresponding to the Linkset.

7.1.8. Create a new Link

Using CLI

You can create a new Link by issuing the command `linkset link create` with appropriate parameters as described below:

```
Name
    linkset link create

SYNOPSIS
    linkset link create span <span-num> code <code-num> channel <channel-num>
    <linkset-name> <link-name>

DESCRIPTION
    This command is used to create a new Link within a Linkset. The Linkset must be
    created prior to executing this command.

PARAMETERS
    span <span-num>   -   Port number in the Card (indexed from 0).
                    Takes an Integer Value.

    code <code-num>       -   Signaling Link code
                    SLS (Signaling link selection) assigned to this
                    Link. Takes an Integer Value.

    channel <channel-num>     -   Time Slot number (TDM time slot).
                    Takes an Integer Value.

    <linkset-name>        -   Name of the Linkset within which the new Link is
                    being created.

    <link-name>       -   Name of the Link to be created. This will be used
                    to identify the Link. Takes a String value.

EXAMPLES
    linkset link create span 1 code 1 channel 1 linkset1 link1

    The above command will create a new Link identified as link1 within an existing
    Linkset identified as linkset1.
```

Using GUI

Procedure: Create new Link using GUI

1. In the section for Links in the Linkset Management Unit window, click on the 'Create Link' button. This will launch a pop-up 'Create Link'.
2. In the 'Create Link' pop-up, add details of the new Link. You must ensure that you fill in all the parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new Link will be created with parameters as specified. If there is an error in creating the Link then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Link' pop-up.

7.1.9. Remove a Link

Using CLI

You can delete an existing Link by issuing the command `linkset link delete` with appropriate parameters as described below:

```
Name
    linkset link delete

SYNOPSIS
    linkset link delete <linkset-name> <link-name>

DESCRIPTION
    This command is used to delete an existing Link within a Linkset.

PARAMETERS
    <link-name>       -   Name of the Link to be deleted.

    <linkset-name>        -   Name of the Linkset within which the Link resides.

EXAMPLE
    linkset link delete linkset1 link1

    The above command will delete the Link identified by the name Link1 within the
    Linkset linkset1.
```

Using GUI

Procedure: Delete Link using GUI

1. Navigate to the 'Links' section in the Linkset Management Unit window and locate the row corresponding to the Link you wish to delete.
2. You must ensure that the Link is deactivated prior to deletion. If the Link is inactive, the last column for 'Delete' will display a 'x' button in red. If the Link is currently active, the 'x' button will be displayed in orange. You can only delete the Link if it is inactive and the 'x' button is displayed in red.
3. Click on the red 'x' button to delete the corresponding Link instance.

7.1.10. Activate Link

Using CLI

You can activate an existing Link by issuing the command `linkset link activate` with appropriate parameters as described below:

```
Name
    linkset link activate

SYNOPSIS
    linkset link activate <linkset-name> <link-name>

DESCRIPTION
    This command is used to activate an existing Link within a Linkset.

PARAMETERS
    <link-name>       -   Name of the Link to be activated.

    <linkset-name>        -   Name of the Linkset within which the Link resides.

EXAMPLE
    linkset link activate linkset1 link1

    The above command will activate the Link identified by the name Link1 within the
    Linkset linkset1.
```

Using GUI

Procedure: Activate Link using GUI

1. Navigate to the 'Links' section in the Linkset Management Unit window and locate the row corresponding to the Link you wish to activate.
2. Click on the 'Activate' button to activate that Link within the Linkset.

7.1.11. Deactivate Link

Using CLI

You can deactivate a currently active Link by issuing the command `linkset link deactivate` with appropriate parameters as described below:

```
Name
    linkset link deactivate

SYNOPSIS
    linkset link deactivate <linkset-name> <link-name>

DESCRIPTION
    This command is used to deactivate an existing Link within a Linkset.

PARAMETERS
    <link-name>       -   Name of the Link to be deactivated.

    <linkset-name>        -   Name of the Linkset within which the Link resides.

EXAMPLE
    linkset link deactivate linkset1 link1

    The above command will deactivate the Link identified by the name Link1 within the
    Linkset linkset1.
```

Using GUI

Procedure: De-activate Link using GUI

1. Navigate to the 'Links' section in the Linkset Management Unit window and locate the row corresponding to the Link you wish to deactivate.
2. Click on the 'Deactivate' button to deactivate that Link within the Linkset.

7.2. SCTP Management

7.2.1. Using CLI

You can manage all SCTP related configurations through the Command Line Interface by using the `sctp` command. You can create, destroy, start and stop SCTP Servers / Associations by issuing the `sctp` command with appropriate parameters.

7.2.2. Using GUI

The GUI will allow you to manage your SCTP Servers and Associations efficiently using a user-friendly interface. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'SCTP' link in the left panel. The main panel will display the names of all configured SCTP Management units. To configure or view the settings of a particular SCTP Management Unit you must click on the name of that unit. The GUI will look similar to the figure below and is divided into three tabs.

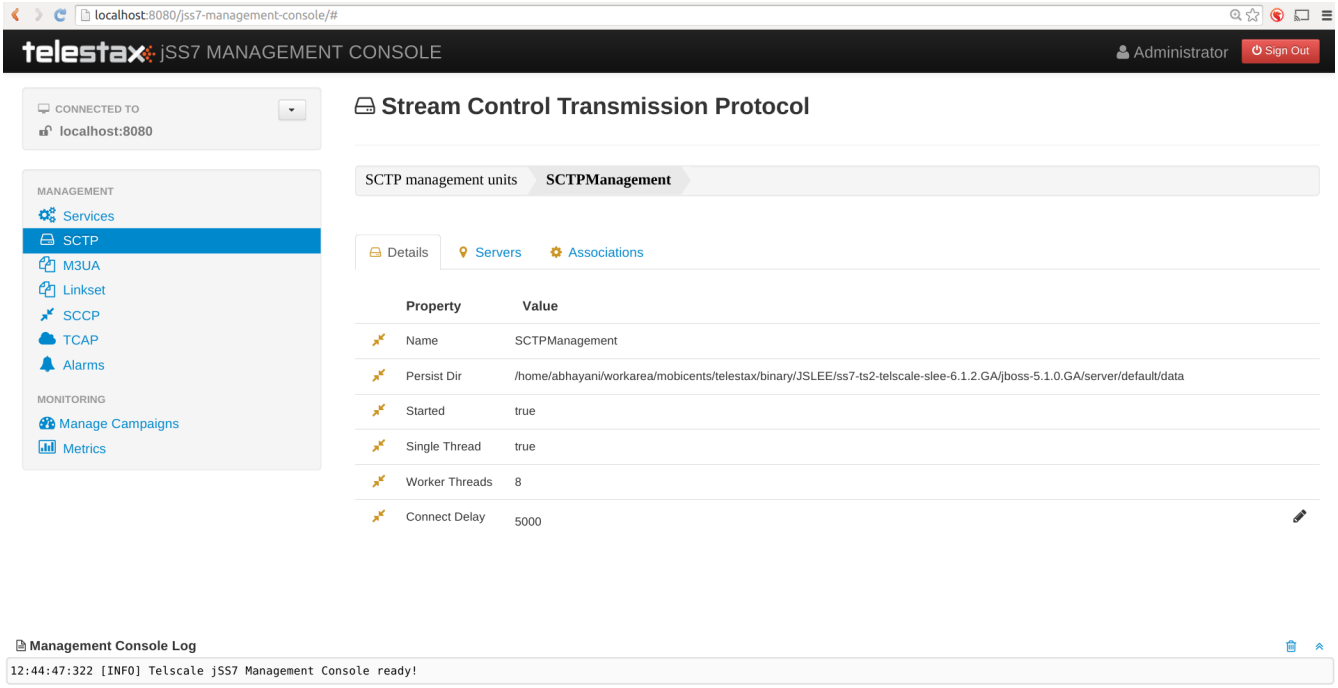


Figure 13. GUI - SCTP Management

The first tab will display the properties of the SCTP Management unit. These details displayed here are fetched from the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name. These properties can be modified here in the GUI. To modify them you must click the pencil, change the value and save. The GUI will then display the modified values.

The other two tabs will allow you to manage and monitor all Servers and Associations within this SCTP Management unit.

7.2.3. SCTP stack properties

Connect Delay

Using CLI

You can set the 'Connect Delay (milliseconds)' by issuing the command `sctp set connectdelay` with appropriate parameters as described below. You can verify this by issuing the command `sctp get connectdelay` which will display the value set for this property.

```
Name
    sctp set connectdelay

SYNOPSIS
    sctp set connectdelay <connectdelay> stackname <stack-name>

DESCRIPTION
    If the SCTP Socket is client-side, connectDelay specifies
    the delay time in milliseconds after which a connection
    with the server will be attempted. This delay is necessary
    when there is network disruption and the connection between
    the client and the server breaks, so that the SCTP stack
    doesn't continuously attempt to reconnect.
    Defalut is 30000 milliseconds.

PARAMETERS

    Standard Parameters

    <connectdelay>    -   Connect delay in milliseconds.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sctp set connectdelay 40000
```

Using GUI

On SCTP management page, click on pencil against the 'Connect Delay' property and text box becomes editable. Change value and save.

Single Thread

You can modify the settings for the parameter 'singlethread' only when the SCTP Stack is not running. In addition, this parameter cannot be modified through the CLI or GUI. You will have to invoke the setter function directly from the source code.

If you are using the JBoss Application Server, then you can set this parameter by adding a property (as shown below) to the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name.

```
/*Add property for the parameter 'singleThread' to jboss-beans.xml file and specify
settings to true or false*/
<property name="singleThread">true</property>
```

The current settings of the parameter can be viewed in the GUI or by invoking the appropriate CLI command as described below.

Using CLI

You can retrieve the current settings of the parameter 'singlethread' by issuing the command `sctp get singlethread`. However as explained above, you cannot modify the settings through the CLI.

```
Name
    sctp get singlethread

SYNOPSIS
    sctp get singlethread

DESCRIPTION
    This command is used to retrieve the current settings of the parameter
    'singlethread'. The 'singlethread' parameter is used to specify if there
    can be more than one worker threads dedicated to call the applications
    above SCTP.

    The default settings will dedicate only one thread for calling applications
    above SCTP. If you wish to assign multiple worker threads, then the value of
    the paramter 'singlethread' must be set to false before you can change the
    number of worker threads to more than one.

    The settings can be modified only when the SCTP Stack is not running.
    To modify this parameter you must invoke the setter function
    directly from the code or if you are using the JBoss AS, you can add a
    property to the XML descriptor file jboss-beans.xml. You cannot change the
    settings through the CLI.
```

Using GUI

In the SCTP management page, you can view the current settings of the 'Single Thread' property. But as explained above, you cannot change the settings in the GUI. But as explained above, you cannot change the settings in the GUI. For more details about this parameter, refer to the detailed description about the parameter in the above section for CLI.

Worker Threads

You can modify the settings for the parameter 'workerthreads' only when the SCTP Stack is not running. In addition, this parameter cannot be modified through the CLI or GUI. You will have to invoke the setter function directly from the source code.

If you are using the JBoss Application Server, then you can set this parameter by adding a property (as shown below) to the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name.

```
/*Add property for the parameter 'workerthreads' to jboss-beans.xml file and specify
the value*/
<property name="workerThreads">4</property>
```

The current settings of the parameter can be viewed in the GUI or by invoking the appropriate CLI command as described below.

Using CLI

You can retrieve the current settings of the parameter 'workerthreads' by issuing the command `sctp get workerthreads`. However as explained above, you cannot modify the settings through the CLI.

```
Name
    sctp get workerthreads

SYNOPSIS
    sctp get workerthreads

DESCRIPTION
    This command is used to retrieve the current settings of the parameter
    'workerthreads'. The 'workerthreads' parameter is used to specify the number
    of worker threads dedicated to call the applications above SCTP.

    The default settings will dedicate only one thread for I/O and one thread
    for calling applications above SCTP. If you wish to assign multiple
    worker threads, then the value of the paramter 'singlethread' must be
    set to false and the number of worker threads must be set using this parameter
    'workerthreads'.

    The settings can be modified only when the SCTP Stack is not running.
    To modify this parameter you must invoke the setter function
    directly from the code or if you are using the JBoss AS, you can add a
    property to the XML descriptor file jboss-beans.xml. You cannot change the
    settings through the CLI.
```

Using GUI

In the SCTP management page, you can view the current settings of the 'Worker Threads' property. But as explained above, you cannot change the settings in the GUI. For more details about this parameter, refer to the detailed description about the parameter in the above section for CLI.

7.2.4. View all SCTP (or TCP) Server Instances

Using CLI

You can view the details of all configured SCTP (or TCP) Server instances by issuing the command `sctp server show` as described below:

```
Name
    sctp server show

SYNOPSIS
    sctp server show stackname <stack-name> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all SCTP Server instances created.
    The information displayed will include the socket type (SCTP or TCP), name of the
    Server, state (whether started=false or true), the IP address and port that the
    Server is bound to. For multi-home SCTP Servers it will display all the
    IP addresses that are configured.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific SCTP Management unit and switch to the 'Servers' tab. Here you can view a list of all the Servers created. Every correctly configured Server will be displayed in a row and for each Server, the first column will display the name of the Server. The icon adjacent to the name will be lit 'green' if the server is currently running or if the server is stopped the icon will be lit 'orange'. The second column will indicate the current state of the Server (Started / Stopped), the third column will allow you to Start / Stop the Server and the fourth column will allow you to delete the Server.

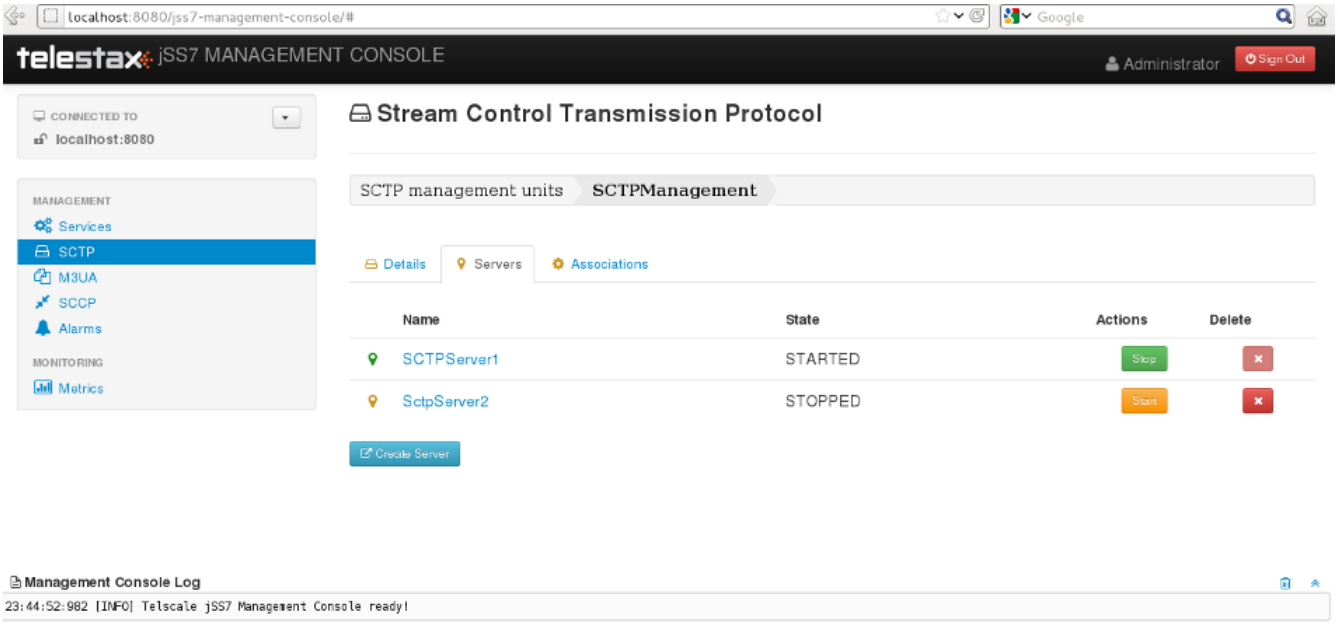


Figure 14. GUI - SCTP Management - Servers

In the screen above, click on the name of the Server whose details you wish to view. This will launch the Server Details and display all the configured properties of the selected Server. The second tab in this view will allow you to view all Associations linked to this particular Server. You can click on any Association name here to view the configured properties. You can click on the bread crumbs at the top to return to any of the previous pages you navigated through.

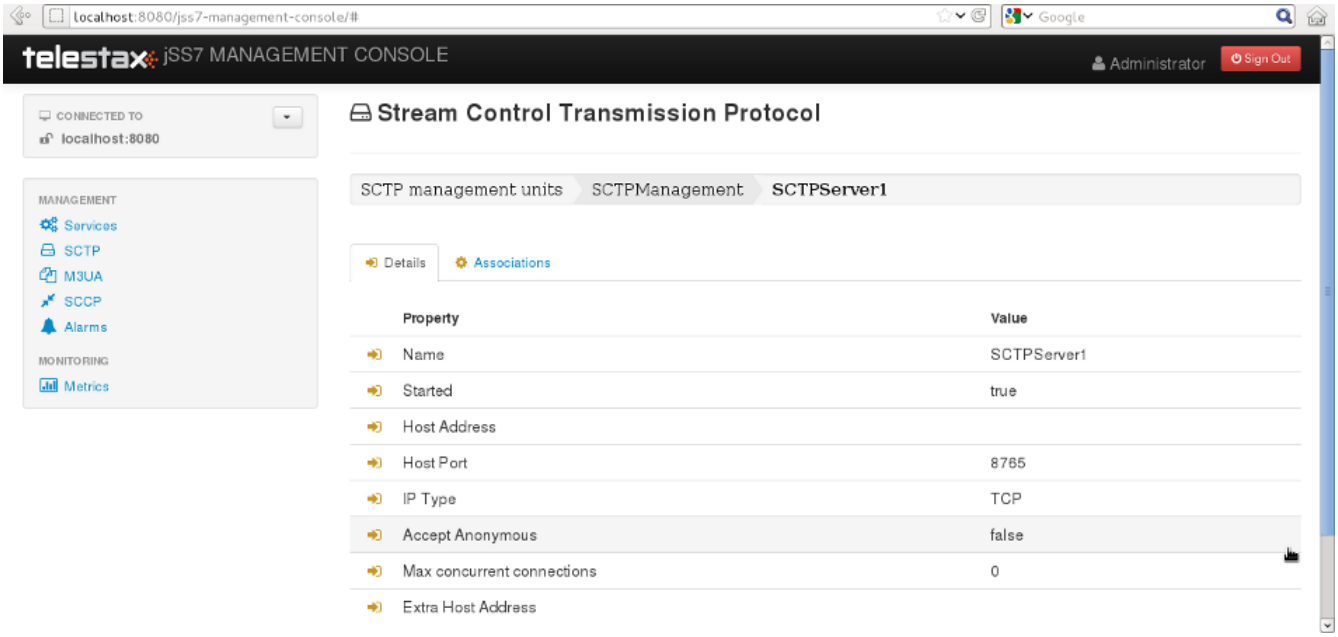


Figure 15. GUI - SCTP Management - Server Details

7.2.5. Create a new SCTP (or TCP) Server Instance

Using CLI

You can create a new SCTP Server by issuing the command `sctp server create` with appropriate parameters as described below:

```
Name
    sctp server create

SYNOPSIS
    sctp server create <sever-name> <host-ip> <host-port> <socket-type> stackname
<stack-name>

DESCRIPTION
    This command is used to create a new SCTP Server (or TCP Server) instance.

PARAMETERS

    Standard Parameters

    <server-name> -   Name of the new Server created. This must be
                    unique and takes any String value.

    <host-ip>     -   The host IP address to which the SCTP Server
                    socket will bind to.

                    For SCTP multi-home support, you can pass multiple
                    IP addresses as comma separated values. The Server
                    socket will bind to the primary IP address and
                    when it becomes unavailable, it will automatically
                    fall back to secondary address. If the socket-type
                    is TCP, these comma separated values will be
                    ignored and the Server socket will always bind to
                    the primary IP address (the first value in the
                    comma separated list).

    <host-port>       -   The host port to which the underlying SCTP Server
                    socket will bind to.

    Optional Parameters

    <socket-type> -   If you do not specify the socket-type as "TCP",
                    by default it will be SCTP.

    <stack-name>  -   Name of the stack on which this command is executed.
                        If not passed, the first stack configured in ShellExecutor
                        will be used.

EXAMPLES
    sctp server create TestServer 127.0.0.1 2905

    The above command will create a new SCTP Server identified as TestServer and bind
    the Server socket to the IP address 127.0.0.1 and port 2905.

    sctp server create TestServerMulti 10.2.50.145,10.2.50.146 2905

    The above command will create a new SCTP Server identified as TestServerMulti and
```

```
    bind the Server socket to the IP address 10.2.50.145 and port 2905. If 10.2.50.145
    is unavailable, the Server will automatically fall back to 10.2.50.146.

    sctp server create TestServerTCP 127.0.0.1 2906 TCP

    The above command will create a new TCP Server identified as TestServerTCP and
    bind the socket to the IP address 127.0.0.1 and port 2906.
```

Using GUI

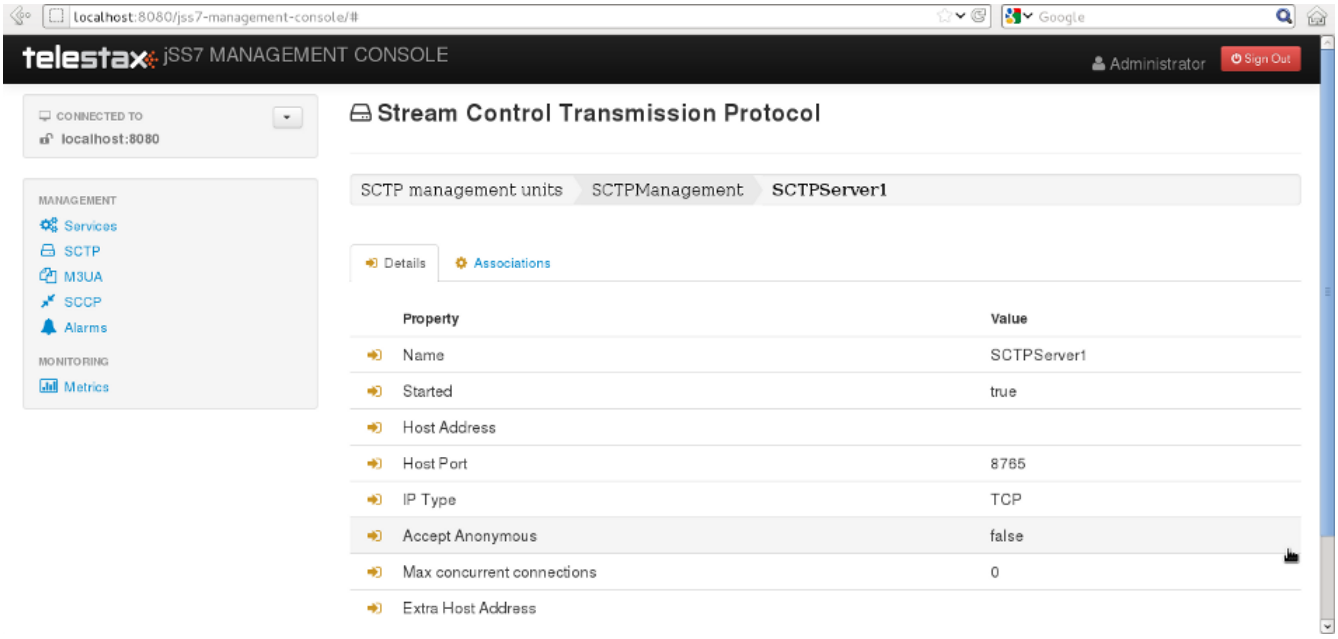


Figure 16. GUI - SCTP Management - Server Create

Procedure: Create new SCTP Server (or TCP Server) instance using GUI

1. In the section for Servers in the SCTP Management Unit window, click on the 'Create Server' button. This will launch a pop-up 'Create Server'.
2. In the 'Create Server' pop-up, add details of the new Server. You must ensure that you fill in all the mandatory parameters (Name, Host Address, Host Port, IP Type, Max Concurrent Connections). For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new SCTP Server (or TCP Server) will be created with parameters as specified. If there is an error in creating the Server then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Server' pop-up.

7.2.6. Delete a SCTP (or TCP) Server Instance

Using CLI

You can delete an existing SCTP Server by issuing the command `sctp server destroy` with appropriate parameters as described below:

```
Name
    sctp server destroy

SYNOPSIS
    sctp server destroy <sever-name> stackname <stack-name>

DESCRIPTION
    This command is used to delete an existing SCTP Server instance. You must ensure
    that the Server is stopped prior to deletion.

PARAMETERS

    Standard Parameters

    <server-name>     -   Name of the Server instance to be deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                        If not passed, the first stack configured in ShellExecutor
                        will be used.

EXAMPLES
    sctp server destroy TestServer

    The above command will destroy the Server identified by the name TestServer.
```

Using GUI

Procedure: Delete SCTP Server (or TCP Server) instance using GUI

1. Navigate to the 'Servers' section in the SCTP Management Unit window and locate the row corresponding to the Server you wish to delete.
2. You must ensure that the Server is stopped prior to deletion. If the Server is stopped, the last column for 'Delete' will display a 'x' button in red and will be enabled. If the Server is currently running, the 'x' button will be disabled. You can only delete the server if it is stopped.
3. Click on the red 'x' button to delete the corresponding Server instance.

7.2.7. Start a SCTP (or TCP) Server Instance

Using CLI

You can start an existing SCTP Server by issuing the command `sctp server start` with appropriate parameters as described below:

```
Name
    sctp server start

SYNOPSIS
    sctp server start <sever-name> stackname <stack-name>

DESCRIPTION
    This command is used to start an existing SCTP Server instance. Upon executing
    this command, the underlying SCTP server socket is bound to the IP: Port
    configured for this Server instance at the time of creation using the
    "sctp server create" command.

PARAMETERS

    Standard Parameters

    <server-name> -   Name of the Server instance to be started.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                        If not passed, the first stack configured in ShellExecutor
                        will be used.

EXAMPLES
    sctp server start TestServer

    The above command will start the previously created Server instance identified
    by the name TestServer and bind the underlying socket to the IP address and port
    configured for TestServer at the time of creation.
```

Using GUI

Procedure: Start a SCTP Server (or TCP Server) instance using GUI

1. Navigate to the 'Servers' section in the SCTP Management Unit window and locate the row corresponding to the Server you wish to start.
2. Click on the 'Start' button in the actions column to start the corresponding Server instance. The SCTP Server will be started and the underlying SCTP server socket will be bound to the IP: Port configured for this Server instance at the time of creation.
3. If the Server has started successfully you will find the status indicating the Server as 'Started' and the Server's icon will be lit green. If there is an error and the Server failed to start, you will find details of the error in the Management Console log below.

7.2.8. Stop a SCTP (or TCP) Server Instance

Using CLI

You can stop a currently running SCTP Server by issuing the command `sctp server stop` with

appropriate parameters as described below:

```
Name
    sctp server stop

SYNOPSIS
    sctp server stop <sever-name> stackname <stack-name>

DESCRIPTION
    This command is used to stop an existing SCTP Server instance. Upon executing this
    command, the underlying SCTP server socket is closed and all resources are
    released.

PARAMETERS

    Standard Parameters

    <server-name>     -   Name of the Server instance to be stopped.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                        If not passed, the first stack configured in ShellExecutor
                        will be used.

EXAMPLES
    sctp server stop TestServer

    The above command will stop the currently running Server instance identified by
    the name TestServer, close the underlying socket and release all resources.
```

Using GUI

Procedure: Stop a SCTP Server (or TCP Server) instance using GUI

1. Navigate to the 'Servers' section in the SCTP Management Unit window and locate the row corresponding to the Server you wish to stop.
2. To stop a Server currently running, click on the 'Stop' button in the actions column of the row corresponding to the Server instance. When the Server is stopped the underlying SCTP server socket will be closed and all resources are released.

7.2.9. View all SCTP (or TCP) Associations

Using CLI

You can view the details of all configured SCTP (or TCP) Associations by issuing the command `sctp association show` as described below:

```
Name
    sctp association show

SYNOPSIS
    sctp association show stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all SCTP Associations created.
    The information displayed will include the Association type (SERVER or CLIENT),
    name of the Association, state (whether started=false or true). For a CLIENT
    Association it will also display the host-ip, host-port and peer-ip, peer-port
    values.

    For multi-home SCTP, it will display all the IP addresses that are configured.
    For a SERVER Association, it will display the configured peer-ip and peer-port
    values.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCTP Management unit and switch to the 'Associations' tab. Here you can view a list of all the Associations created. Every correctly configured Association will be displayed in a row and for each Association, the first column will display the name of the Association. The icon adjacent to the name will be lit 'green' if the Association is currently running or if the Association is stopped the icon will be 'orange'. The second column will indicate the current state of the Association (Started / Stopped) and the third column will allow you to delete the Association.

You cannot start or stop a SCTP Association here in this window. Every SCTP Association must be associated with an ASP (M3UA) and will automatically start or stop when the associated ASP is started or stopped. For more details on how to associate with an ASP, please refer to M3UA Management.

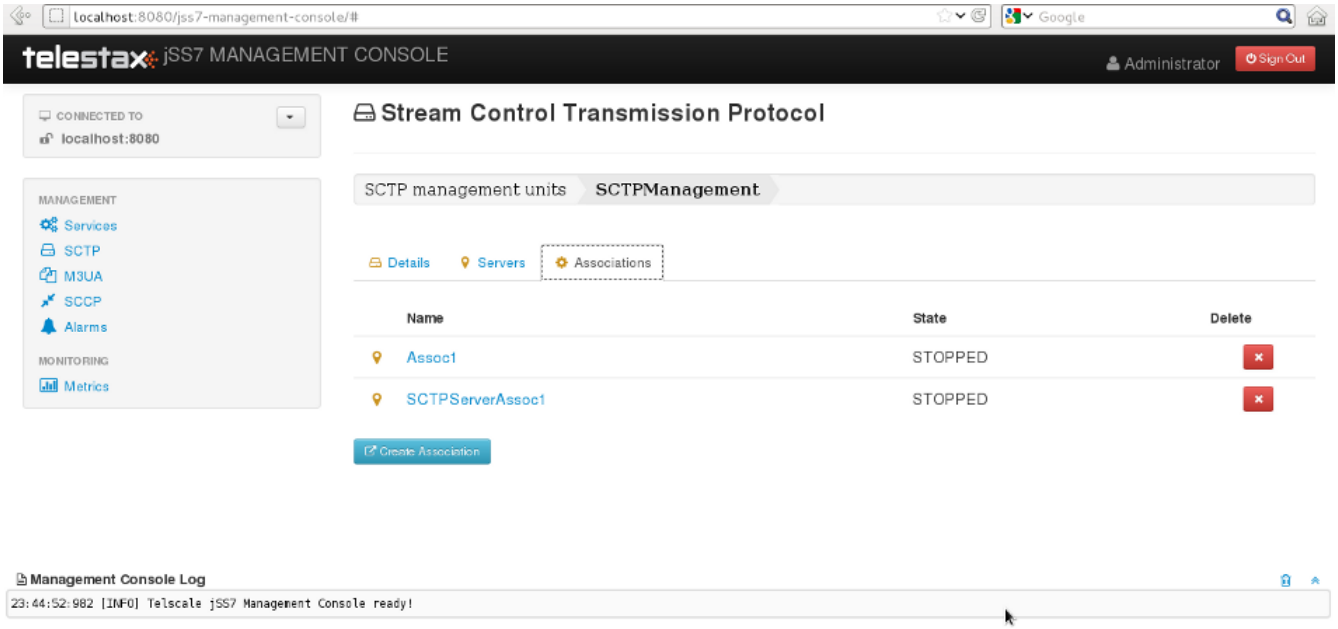


Figure 17. GUI - SCTP Management - Associations

In the screen above, click on the name of the Association whose details you wish to view. This will launch the Association Details and display all the configured properties of the selected Association.

7.2.10. Create a new SCTP (or TCP) Association

Using CLI

You can create a new SCTP Association by issuing the command `sctp association create` with appropriate parameters as described below:

```
Name
    sctp association create

SYNOPSIS
    sctp association create <assoc-name> <CLIENT | SERVER> <server-name> <peer-ip>
    <peer-port> <host- ip> <host-port> <socket-type> stackname <stack-name>

DESCRIPTION
    This command is used to create a new SCTP (Client side or Server side)
    association.

PARAMETERS

    Standard Parameters

    <assoc-name>      -   Name of the new Association created. This must be
                    unique and takes any String value.

    <CLIENT | SERVER> -   Specify if this association is client side or
                    server side. If it is client side, it will
                    initiate the connection to peer. If it is server
                    side, it will wait for peer to initiate the
```

```
                connection. The connection request will be
                accepted from peer-ip: peer:port.

<peer-ip>     -   In a client side association, the server IP
                address to which the client is connecting to.

                In a server side association, the client IP
                address from which connections will be accepted.

<peer-port>       -   In a client side association, the server Port
                to which the client is connecting to.

                In a server side association, the client port from
                which connections will be accepted.

<host-ip>     -   In a client side association, the local IP address
                to which the socket will bind to.

                For SCTP multi-home support, you can pass multiple
                IP addresses as comma separated values. The
                Association will bind to the primary IP address
                and when it becomes unavailable, it will
                automatically fall back to secondary address.
                If the socket-type is TCP, these comma separated
                values will be ignored and the Assocation will
                always bind to the primary IP address (the first
                value in the comma separated list). This is
                required only for a client side Association.

                For a server side association, even if you specify
                these values it will be ignored.

<host-port>       -   In a client side association, the local port to
                which the socket will bind to. This is
                required only for a client side Association.

                For a server side association, even if you specify
                these values it will be ignored.

<server-name> -   In a server-side association, the server-name must
                be passed to associate with the Server identified
                by that name. You must ensure that the Server
                identified by server-name has already been created
                using the sctp server create command.

                In a client-side association, this is not required
                and you should not pass this parameter.

Optional Parameters

<socket-type> -   If you do not specify the socket-type as "TCP",
```

```
                    by default it will be SCTP. If it is a
                    SERVER SCTP Association, the socket-type must
                    match with the one specified while creating the
                    Server.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sctp association create Assoc1 CLIENT
    192.168.56.101 2905 192.168.56.1,192.168.56.1 2905

    The above command will create a new CLIENT SCTP Association identified as Assoc1.
    The client side will initiate the connection. It will bind the
    host-ip 192.168.56.1 and host-port 2905 to the Server IP 192.168.56.101 and
    port 2905.

    sctp association create Assoc2 SERVER TestServer 192.168.56.1 2905

    The above command will create a new SERVER SCTP association with the Server
    identified as TestServer and accept connections from peer whose IP address is
    192.168.56.1 and port 2905.
```

Using GUI

Procedure: Create new SCTP (or TCP) Association (Client side or Server side)

1. In the section for Associations in the SCTP Management Unit window, click on the 'Create Association' button. This will launch a pop-up 'Create Association'.
2. In the 'Create Association' pop-up, add details of the new Association. You must ensure that you fill in all the mandatory parameters: Name, Peer Address, Peer Port, Server Name (for Server side Association), Host Address and Host Port (for Client side Association). For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new SCTP Association (or TCP Association) will be created with parameters as specified. If there is an error in creating the Association then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Association' pop-up.

7.2.11. Delete a SCTP (or TCP) Association

Using CLI

You can delete an existing SCTP Association by issuing the command `sctp association destroy` as described below:

```
Name
    sctp association destroy

SYNOPSIS
    sctp association destroy <assoc-name> stackname <stack-name>

DESCRIPTION
    This command is used to delete an existing SCTP Association identified by the
    name assoc-name.

PARAMETERS

    Standard Parameters

    <assoc-name>  -   Name of the Association to be deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sctp association destroy Assoc1

    The above command will destroy the Association identified by the name Assoc1.
```

Using GUI

Procedure: Delete SCTP Association (or TCP Association) instance

1. Navigate to the 'Associations' section in the SCTP Management Unit window and locate the row corresponding to the Association you wish to delete.
2. You must ensure that the Association is stopped prior to deletion. If the Association is stopped, the last column for 'Delete' will display a 'x' button in red. If the Association is currently running, the 'x' button will be displayed in orange. You can only delete the Association if it is stopped and the 'x' button is displayed in red.
3. Click on the red 'x' button to delete the corresponding Association instance.

7.3. M3UA Management

7.3.1. Using CLI

You can manage all M3UA Application Server, Application Server Process and Route related configurations through the Command Line Interface by using the `m3ua` command. You can create, destroy, start and stop ASPs by issuing the `m3ua asp` command with appropriate parameters. You can create, destroy, add and remove AS by issuing the `m3ua as` command with appropriate parameters. You can add and remove M3UA Routes by issuing the `m3ua route` command with appropriate

parameters.

7.3.2. Using GUI

The GUI will allow you to manage your M3UA configurations efficiently using a user-friendly interface. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'M3UA' link in the left panel. The main panel will display the names of all configured M3UA Management units. To configure or view the settings of a particular M3UA Management Unit you must click on the name of that unit. The GUI will look similar to the figure below and is divided into four tabs.

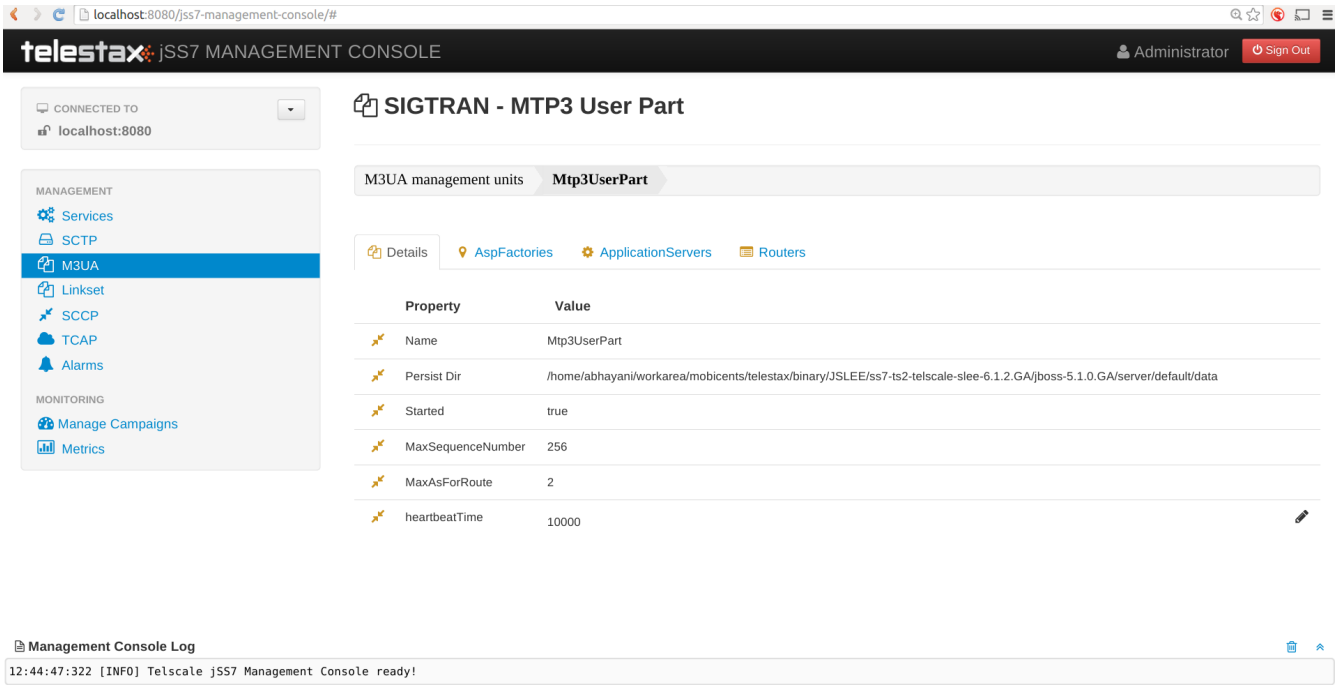


Figure 18. GUI - M3UA Management

The first tab will display the properties of the M3UA Management unit. These details displayed here are fetched from the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name. These properties can be modified here in the GUI. To modify them you must click the pencil, change value and save. The GUI will then display the modified values.

The other three tabs will allow you to manage and monitor all Servers, ASPs and Routers within this M3UA Management unit.

7.3.3. M3UA stack properties

Maximum Sequence Number

You can modify the settings for the parameter 'maxsequencenumber' only when the M3UA Stack is not running. In addition, this parameter cannot be modified through the CLI or GUI. You will have to invoke the setter function directly from the source code.

If you are using the JBoss Application Server, then you can set this parameter by adding a property (as shown below) to the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is

the server profile name.

```
/*Add property for the parameter 'maxsequencenumber' to jboss-beans.xml file and
specify the value*/
<property name="maxSequenceNumber">128</property>
```

The current settings of the parameter can be viewed in the GUI or by invoking the appropriate CLI command as described below.

Using CLI

You can retrieve the current settings of the parameter 'maxsequencenumber' by issuing the command `sctp get maxsequencenumber`. However as explained above, you cannot modify the settings through the CLI.

```
Name
    m3ua get maxsequencenumber

SYNOPSIS
    m3ua get maxsequencenumber

DESCRIPTION
    This command is used to retrieve the current settings of the parameter
    'maxsequencenumber'. The 'maxsequencenumber' parameter is used to specify
    the maximum sequence number used for load-balancing algorithm.

    Sequence number or Signalling Link Selection (SLS) is used for load-balancing
    between ASPs in AS and also between various AS for the same point-code.
    The parameter 'maxsequencenumber' controls the maximum SLS that should be
    used for this. It is safe to leave it at 256.

    The settings can be modified only when the M3UA Stack is not running.
    To modify this parameter you must invoke the setter function
    directly from the code or if you are using the JBoss AS, you can add a
    property to the XML descriptor file jboss-beans.xml. You cannot change the
    settings through the CLI.
```

Using GUI

In the M3UA management page, you can view the current settings of the 'Max Sequence Number' property. But as explained above, you cannot change the settings in the GUI. For more details about this parameter, refer to the detailed description about the parameter in the above section for CLI.

Maximum AS for route

You can modify the settings for the parameter 'maxasforroute' only when the M3UA Stack is not running. In addition, this parameter cannot be modified through the CLI or GUI. You will have to invoke the setter function directly from the source code.

If you are using the JBoss Application Server, then you can set this parameter by adding a property (as shown below) to the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name.

```
/*Add property for the parameter 'maxasforroute' to jboss-beans.xml file and specify
the value*/
<property name="maxAsForRoute">4</property>
```

The current settings of the parameter can be viewed in the GUI or by invoking the appropriate CLI command as described below.

Using CLI

You can retrieve the current settings of the parameter 'maxasforroute' by issuing the command `sctp get maxasforroute`. However as explained above, you cannot modify the settings through the CLI.

```
Name
    m3ua get maxasforroute

SYNOPSIS
    m3ua get maxasforroute

DESCRIPTION
    This command is used to retrieve the current settings of the parameter
    'maxasforroute'. The 'maxasforroute' parameter is used to specify
    the maximum routes for destination point code.

    Every destination point code should be configured in M3UA with the
    corresponding AS. The parameter 'maxasforroute' controls the
    maximum number of AS that can be used to route the message to the
    same Destination Point Code.

    It is better to always maintain this parameter as an even number for better
    load-sharing and a maximum of 2 is standard and widely accepted. You should
    not change this value if there is at least one route defined, else it will
    throw Exception for that route. You have to delete all the routes,
    change this property and redefine routes.

    The settings can be modified only when the M3UA Stack is not running.
    To modify this parameter you must invoke the setter function
    directly from the code or if you are using the JBoss AS, you can add a
    property to the XML descriptor file jboss-beans.xml. You cannot change the
    settings through the CLI.
```

Using GUI

In the M3UA management page, you can view the current settings of the 'Max As for Route'

property. But as explained above, you cannot change the settings in the GUI. For more details about this parameter, refer to the detailed description about the parameter in the above section for CLI.

Heart Beat time

Using CLI

You can set the 'heartbeattime' by issuing the command `m3ua set heartbeattime` with appropriate parameters as described below. You can verify this by issuing the command `m3ua get heartbeattime` which will display the value set for this property.

```
Name
    m3ua set heartbeattime

SYNOPSIS
    m3ua set heartbeattime <heartbeattime> stackname <stack-name>

DESCRIPTION
    Each ASP can send HEART_BEAT to peer to determine the availability of
    link. If there is no traffic M3UA will initiate heart beat every
    'heartbeatTime' milli seconds. If 3 consecutive HEART_BEAT are
    missed, stack will close and re-initiate connection.

PARAMETERS

    Standard Parameters

    <heartbeattime>   -   Heart Beat time in milliseconds.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua set heartbeattime 30000
```

Using GUI

On M3UA management page, click on pencil against the 'Worker Threads' property and text box becomes editable. Change value and save.

7.3.4. View all M3UA Application Server Processes

Using CLI

You can view the details of all configured M3UA Application Server Processes by issuing the command `m3ua asp show` as described below:

```
Name
    m3ua asp show

SYNOPSIS
    m3ua asp show stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Application Server
    Processes. The information displayed will include the name, the SCTP Association
    name and if it is started or stopped.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific M3UA Management unit and switch to the 'AspFactories' tab. Here you can view a list of all the ASPs created. Every correctly configured ASP will be displayed in a row and for each ASP, the first column will display the name of the ASP. The icon adjacent to the name will be lit 'green' if the ASP is currently running or if the ASP is stopped the icon will be lit 'orange'. The second column will indicate the current state of the ASP (true / false), the third column will allow you to Start / Stop the ASP and the fourth column will allow you to delete the ASP.

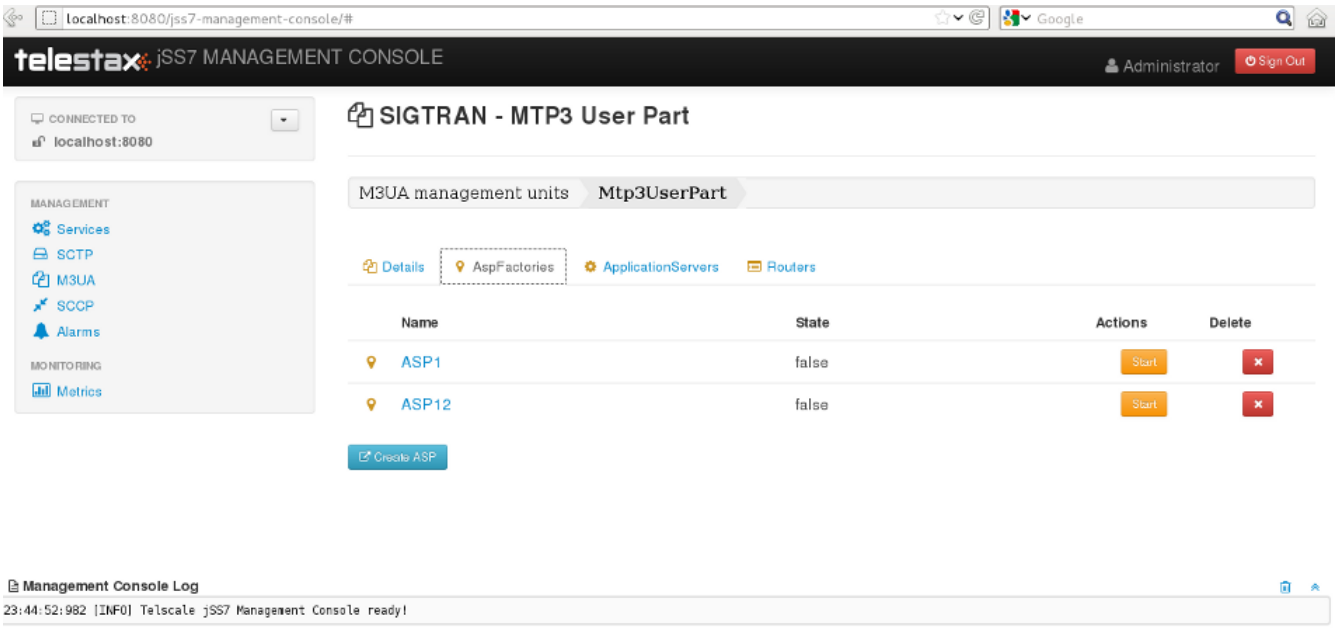


Figure 19. GUI - M3UA Management - AspFactories

In the screen above, click on the name of the ASP whose details you wish to view. This will launch the ASP Details and display all the configured properties of the selected ASP. The second tab in this view will allow you to view all connected Application Servers. You can click on the bread crumbs at

the top to return to any of the previous pages you navigated through.

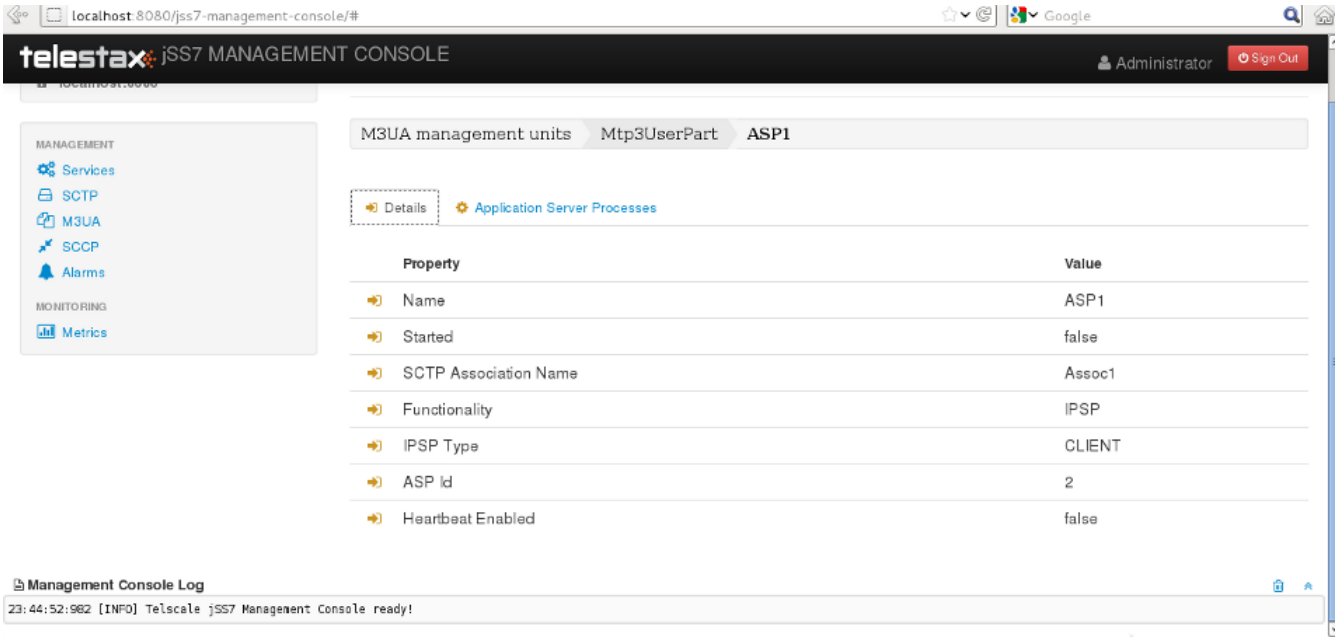


Figure 20. GUI - M3UA Management - ASP Details

7.3.5. Create a new M3UA Application Server Process

Using CLI

You can create a new M3UA ASP by issuing the command `m3ua asp create` with appropriate parameters as described below:

```
Name
    m3ua asp create

SYNOPSIS
    m3ua asp create <asp-name> <sctp-association> aspid <aspid> heartbeat <true|false>
    stackname <stack-name>

DESCRIPTION
    This command is used to create a new Application Server Process.

PARAMETERS

    Standard Parameters

    <asp-name>            -   Name of the new Application Server Process
                          created. This must be unique and takes any
                          String value.

    <sctp-association>        -   Name of the SCTP Association

    Optional Parameters

    <aspid>               -   Identifier for this newly created
```

```
                        Application Server Process. If this is
                        not passed, next available aspid will be
                        used.

    heartbeat <true|false>        -   If this parameter is enabled (value set to
                        true), then heartbeat mechanism is enabled
                        between M3UA peers. When this is enabled,
                        it sends a Heartbeat message every 10
                        seconds. If there is no response for the
                        third heartbeat then it assumes that the
                        underlying network is dead. So it closes
                        the connection and attempts to connect
                        again. The M3UA peers are brought back to
                        the same state as they were prior to
                        dying.

                        This is an optional parameter and if
                        unspecified, heartbeat mechanism is
                        disabled.

    <stack-name>  -   Name of the stack on which this command is executed.
                        If not passed, the first stack configured in ShellExecutor
                        will be used.

EXAMPLES
    m3ua asp create ASP1 Assoc1 aspid 12 heartbeat true

    The above command will create a new M3UA Application Server Process with name ASP1
    and id 12. Heartbeat mechanism is enabled.
```

Using GUI

Procedure: Create a new Application Server Process using GUI

1. Navigate to the tab 'AspFactories' in the M3UA Management Unit window and click on the 'Create ASP' button. This will launch a pop-up 'Create AspFactory'.
2. In the 'Create ASP' page, add details of the new ASP. You must ensure that you fill in all the mandatory parameters (Name, SCTP Association Name). For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section. You must ensure that a correctly configured SCTP Association is created and available prior to creating a new ASP. When the ASP is started or stopped, this corresponding SCTP Association will start / stop automatically.
3. Verify the details entered and then click on the 'Create' button. A new ASP will be created with parameters as specified. If there is an error in creating the ASP then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Server' pop-up.

7.3.6. Delete an Application Server Process

Using CLI

You can delete an existing M3UA ASP by issuing the command `m3ua asp destroy` with appropriate parameters as described below:

```
Name
    m3ua asp destroy

SYNOPSIS
    m3ua asp destroy <asp-name> stackname <stack-name>

DESCRIPTION
    This command is used to delete an existing M3UA Application Server Process
    identified by the name 'asp-name'. You must ensure that the ASP is stopped prior
    to issuing the command.

PARAMETERS

    Standard Parameters

    <asp-name>        -   Name of the ASP to be deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua asp destroy ASP1

    The above command will destroy the ASP identified by the name ASP1.
```

Using GUI

Procedure: Delete an Application Server Process using GUI

1. Navigate to the 'ASPs' section in the M3UA Management Unit window and locate the row corresponding to the ASP you wish to delete.
2. You must ensure that the ASP is stopped and unassigned from AS prior to deletion. If the ASP is stopped, the last column for 'Delete' will display a 'x' button in red and will be enabled. If the Server is currently running, the 'x' button will be disabled.
3. Click on the red 'x' button to delete the corresponding ASP.

7.3.7. Start an Application Server Process

Using CLI

You can start an existing ASP by issuing the command `m3ua asp start` with appropriate parameters as described below:

```
Name
    m3ua asp start

SYNOPSIS
    m3ua asp start <asp-name> stackname <stack-name>

DESCRIPTION
    This command is used to start an existing ASP. You must ensure that the ASP is
    assigned to at least one AS prior to starting it.

PARAMETERS

    Standard Parameters

    <asp-name>        -   Name of the ASP to be started.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua asp start ASP1

    The above command will start the ASP identified by the name ASP1.
```

Using GUI

Procedure: Start an Application Server Process

1. Navigate to the 'AspFactories' tab in the M3UA Management Unit window and locate the row corresponding to the ASP you wish to start.
2. Click on the 'Start' button in the actions column to start the corresponding ASP. You must ensure that the ASP is assigned to at least one AS prior to starting it.
3. If the ASP has started successfully you will find the status indicating the ASP running as 'true' and the icon will be lit green. If there is an error and the ASP failed to start, you will find details of the error in the Management Console log below.

7.3.8. Stop an Application Server Process

Using CLI

You can stop a currently running ASP by issuing the command `m3ua asp stop` with appropriate

parameters as described below:

```
Name
    m3ua asp stop

SYNOPSIS
    m3ua asp stop <asp-name> stackname <stack-name>

DESCRIPTION
    This command is used to stop a currently running ASP.

PARAMETERS

    Standard Parameters

    <asp-name>        -   Name of the ASP to be stopped.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua asp stop ASP1

    The above command will stop the ASP identified by the name ASP1.
```

Using GUI

Procedure: Stop an Application Server Process

1. Navigate to the 'AspFactories' section in the M3UA Management Unit window and locate the row corresponding to the ASP you wish to stop.
2. Click on the 'Stop' button in the actions column to stop the corresponding ASP.
3. If the ASP has stopped successfully you will find the status indicating the ASP running as 'false' and the icon will be lit orange. If there is an error and the ASP failed to stop, you will find details of the error in the Management Console log below.

7.3.9. View all M3UA Application Servers

Using CLI

You can view the details of all configured M3UA Application Servers by issuing the command `m3ua as show` as described below:

```
Name
    m3ua as show

SYNOPSIS
    m3ua as show stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Application Servers.
    The information displayed will include the configured
    functionality (AS or IPSP or SGW), mode (SE or DE), IPSP type (if applicable),
    routing context, traffic mode and network appearance values.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific M3UA Management unit and switch to the 'ApplicationServers' tab. Here you can view a list of all the ASs created. Every correctly configured AS will be displayed in a row and for each AS, the first column will display the name of the AS. The icon adjacent to the name will be lit 'green' if the AS is currently running or if the AS is stopped the icon will be lit 'orange'. The second column will indicate the current state of the AS (defined / undefined), the third column will allow you to delete the AS.

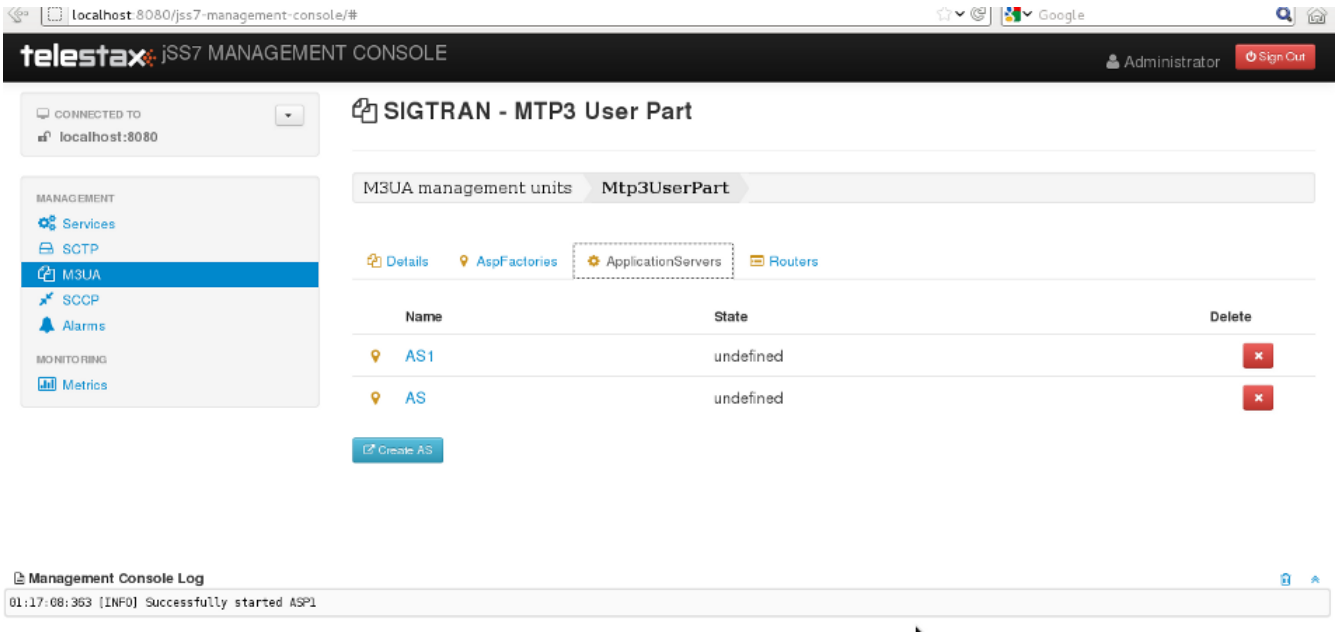


Figure 21. GUI - M3UA Management - ApplicationServers

In the screen above, click on the name of the AS whose details you wish to view. This will launch the AS Details and display all the configured properties of the selected AS. The second tab in this

view will allow you to view the details of the connected ASP. You can click on the bread crumbs at the top to return to any of the previous pages you navigated through.

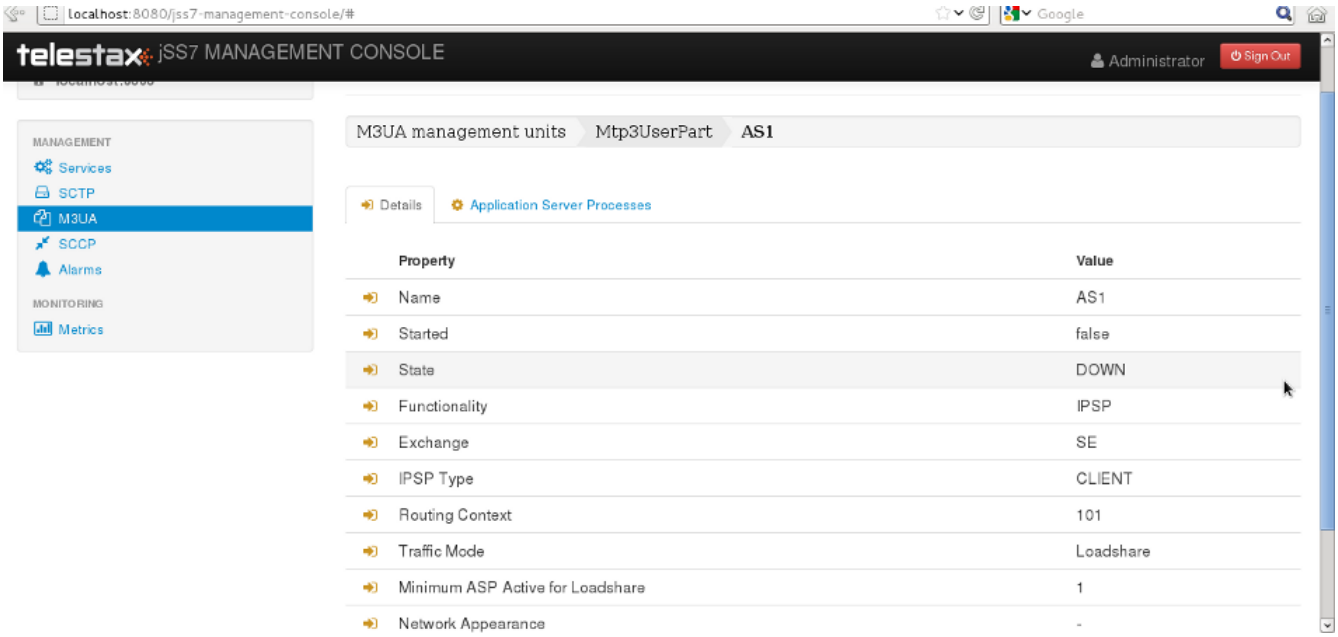


Figure 22. GUI - M3UA Management - AS Details

7.3.10. Create a new M3UA AS

Using CLI

You can create a new M3UA AS by issuing the command `m3ua as create` with appropriate parameters as described below:

```
Name
    m3ua as create

SYNOPSIS
    m3ua as create <as-name> <AS | SGW | IPSP> mode <SE | DE>
    ipspType <client | server> rc <routing- context> traffic-mode <traffic mode>
    min-asp <minimum asp active for TrafficModeType.Loadshare>
    network-appearance <network appearance value> stackname <stack-name>

DESCRIPTION
    This command is used to create a new Application Server.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the new Server created. This must be
                    unique and takes any String value.

    <AS | SGW | IPSP> -   The type of the new Server is specified using this
                      parameter. The three possible values are
                      AS (Application Server), SGW (Signaling Gateway)
```

```
                and IPSP.

    <SE | DE>     -   You must specify if Single Exchange or
                  Double Exchange of ASPSM (ASP State Maintenance)
                  and ASPTM (ASP Traffic Maintenance) messages
                  should be performed.

    <client | server> -   This is required if the newly created AS is of
                  type IPSP. You must specify is if it is of type
                  Client or Server.

    Optional Parameters

    <routing-context> -   This refers to the Routing Context configured for
                  M3UA Stack on SGW. This parameter is optional.

                  However for an ASP (Application Server Process)
                  assigned to this AS to be configured to process
                  signaling traffic related to more than one AS over
                  a single SCTP Association, it is mandatory to
                  specify a routing-context for the AS. If an ASP is
                  configured to always process signaling traffic
                  from one AS, irrespective of whether the received
                  messages have routing context set or not, it will
                  always be delivered to AS for further processing.

                  However if an ASP is configured to process
                  signaling traffic related to more than one AS over
                  a single SCTP Association and if a signaling
                  message is received without RC, then the ASP will
                  drop the message and send back an Error message.
                  A respective log4j error will also be logged.

    <traffic-mode>        -   You may choose to specify the traffic mode for
                  ASPs. At the moment jSS7 M3UA supports only
                  2 modes: loadshare and override. Broadcast mode is
                  not supported.

                  This is an optional parameter and if not specified
                  the default mode is 'loadshare'.

    <min-asp>         -   You may choose to specify the minumum asp
                  active for traffic mode 'loadshare' before the
                  payload starts flowing.

                  This is an optional parameter and if not specified
                  the default value is 1. Also if traffic-mode is not
                  'loadshare' setting this value has no effect.

    <network-appearance>  -   This is a M3UA local reference (typically an
                  integer) shared by SG and AS. This value together
```

```
                    with a Signaling Point Code, uniquely identifies a
                    SS7 node by indicating the specific SS7 network to
                    which it belongs. It can be used to distinguish
                    between signalling traffic, associated with
                    different networks, being sent between the SG and
                    the ASP over a common SCTP association.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua as create AS1 IPSP mode DE ipspType server rc 1 traffic-mode loadshare

    The above command will create a new M3UA Application Server identified as AS1,
    of type IPSP (Server), Double Exchange mode. The Routing Context is 1 and
    traffic-mode is 'loadshare'.

    m3ua as create AS2 AS mode SE rc 100 traffic-mode loadshare 2 network-appearance
12

    The above command will create a new M3UA Application Server identified as AS2,
    of type AS, Single Exchange mode. The Routing Context is 100,
    traffic-mode is 'loadshare' and minimun asp to be active for payload transfer is
2.
    The network-appearance value is 12.
```

Using GUI

Procedure: Create a new M3UA Application Server using GUI

1. Navigate to the tab 'ApplicationServers' in the M3UA Management Unit window and click on the 'Create AS' button. This will launch a pop-up 'Create AS'.
2. In the 'Create Application Server' pop-up, add details of the new AS. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new AS will be created with parameters as specified. If there is an error in creating the AS then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Application Server' pop-up.

7.3.11. Delete a M3UA AS

Using CLI

You can create a new M3UA AS by issuing the command `m3ua as create` with appropriate parameters as described below:

```
Name
    m3ua as destroy

SYNOPSIS
    m3ua as destroy <as-name> stackname <stack-name>

DESCRIPTION
    This command is used to delete an existing M3UA Application Server instance
    identified by the name 'as-name'. You must ensure that all ASPs are unassigned and
    the AS state is 'INACTIVE' prior to destroying the AS.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the AS instance to be deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua as destroy AS1

    The above command will destroy the AS identified by the name AS1.
```

Using GUI

Procedure: Delete a M3UA Application Server using GUI

1. Navigate to the 'ApplicationServers' tab in the M3UA Management Unit window and locate the row corresponding to the AS you wish to delete.
2. You must ensure that all ASPs are unassigned from this AS and the current state of the AS is 'INACTIVE' (displayed as 'undefined') prior to destroying the AS. If the AS is inactive, the last column for 'Delete' will display a 'x' button in red and will be enabled. You can only delete the AS if it is inactive.
3. Click on the red 'x' button to delete the corresponding AS.

7.3.12. Assign an ASP to an AS

Using CLI

You can assign an ASP to an AS by issuing the command `m3ua as add` with appropriate parameters as described below:

```
Name
    m3ua as add

SYNOPSIS
    m3ua as add <as-name> <asp-name> stackname <stack-name>

DESCRIPTION
    This command is used to assign an Application Server Process to an Application
    Server. The AS and ASP must both be created prior to executing this command.

    You can configure an ASP to process signaling traffic related to more than one AS,
    over a single SCTP Association. However you must ensure that all the Application
    Servers that share the ASP are configured with a valid Routing Context value.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the AS to which this ASP is being assigned.

    <asp-name>        -   Name of the ASP that is being assigned to the AS.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua as add AS1 ASP1
    - The above command will assign ASP1 to AS1.
```

Using GUI

Procedure: Assign an ASP to an AS using GUI

1. Navigate to the 'ApplicationServers' tab in the M3UA Management Unit window, locate the row corresponding to the AS you wish to assign an ASP and click on the name of the AS.
2. This will launch the AS details page where all the properties of the AS will be displayed. Switch to the second tab in this view called "Application Server Processes". As shown in the figure below, you will find a list of all currently assigned ASPs to this selected AS.

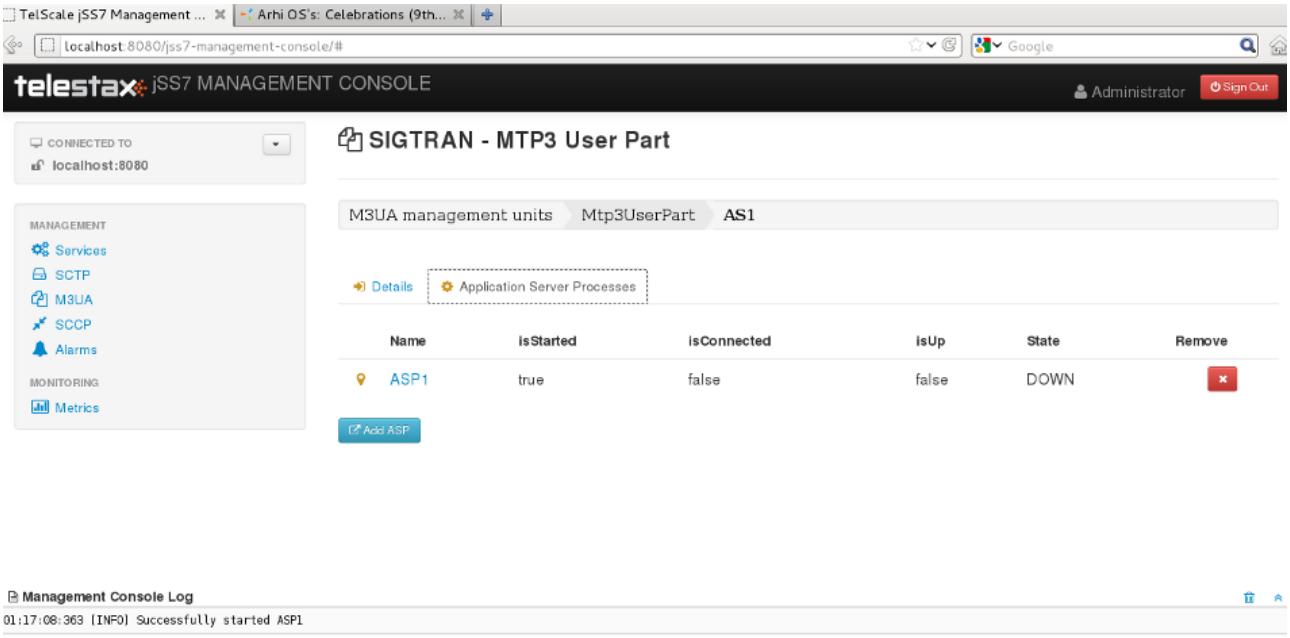


Figure 23. GUI - M3UA Management - Assign ASP to an AS

3. Click on the 'Add ASP' button at the bottom. This will launch a pop-up named 'Add ASP' where all available ASPs will be listed in a drop down box.
4. Click on the 'Create' button to add the selected ASP to this AS. The ASP will be assigned to this AS and will be displayed in the ASP list for this AS.
5. You can configure an ASP to process signaling traffic related to more than one AS, over a single SCTP Association. However you must ensure that all the Application Servers that share the ASP are configured with a valid Routing Context value.

7.3.13. Unassign an ASP from an AS

Using CLI

You can unassign an ASP from an AS by issuing the command `m3ua as remove` with appropriate parameters as described below:

```
Name
    m3ua as remove

SYNOPSIS
    m3ua as remove <as-name> <asp-name> stackname <stack-name>

DESCRIPTION
    This command is used to un-assign an Application Server Process from an
    Application Server that it was previously assigned to.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the AS from which this ASP is being
                    un-assigned.

    <asp-name>        -   Name of the ASP to be un-assigned.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    m3ua as remove AS1 ASP1

    The above command will remove ASP1 from AS1.
```

Using GUI

Procedure: Unassign an ASP from an AS using GUI

1. Navigate to the 'ApplicationServers' tab in the M3UA Management Unit window, locate the row corresponding to the AS you wish to unassign an ASP from and click on the name of the AS.
2. This will launch the AS details page where all the properties of the AS will be displayed. Switch to the second tab in this view called "Application Server Processes". As shown in the figure above, you will find a list of all currently assigned ASPs to this selected AS.
3. Locate the row corresponding to the ASP you wish to unassign from this AS.
4. Click on the red coloured 'x' remove button in the row corresponding to the ASP you wish to remove. This action will unassign the ASP from this AS.

7.3.14. View all M3UA Routes

Using CLI

You can view the details of all configured M3UA Routes by issuing the command `m3ua route show` as

described below:

```
Name
    m3ua route show

SYNOPSIS
    m3ua route show stackname <stack-name>

DESCRIPTION
    This command is used to display all configured routes.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific M3UA Management unit and switch to the 'Routers' tab. Here you can view a list of all the Routes created as shown in the figure below. Every correctly configured Route will be displayed in a row and for each Route, the first column will display DPC:OPC:SI values. The icon adjacent to the name will be lit 'green' if the Route is currently active or if the Route is inactive the icon will be 'orange'. The second column will indicate the current state of the Route (Active / Inactive) and the third column will display the name of the AS assigned to route messages for this DPC.

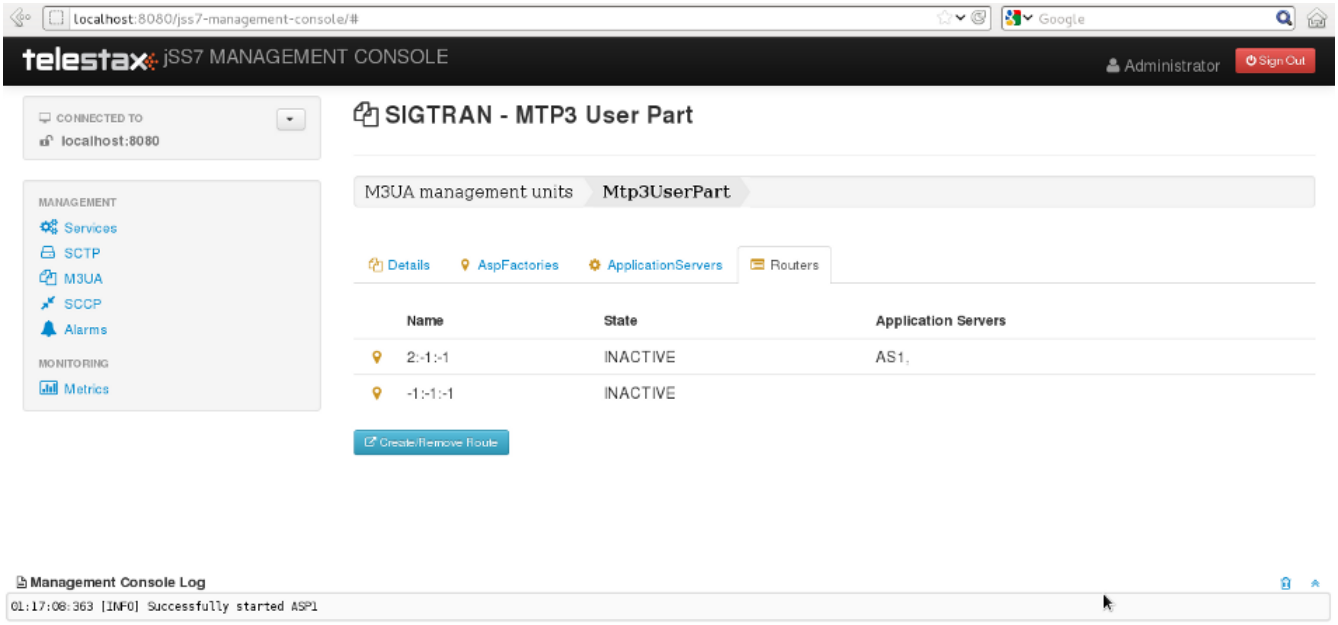


Figure 24. GUI - M3UA Management - Route

7.3.15. Create a new M3UA Route

Using CLI

You can create a new M3UA Route by issuing the command `m3ua route add` with appropriate parameters as described below:

```
Name
    m3ua route add

SYNOPSIS
    m3ua route add <as-name> <dpc> <opc> <si> trafficmode <traffic-mode> stackname
<stack-name>

DESCRIPTION
    This command is used to configure an AS to route message, i.e. configure the
    destination point code that the message will be routed to. You must ensure that
    the AS is created prior to executing this command.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the AS assigned to route message for
                    this dpc.

    <dpc>         -   Destination Point Code.

    <opc>         -   Originating Point Code.

    <si>          -   Service Indicator.

    Optional Parameters
    <traffic-mode>    -   There can be two or more AS defined for each route.
                        The M3UA Stack will do load-balancing between these
                        AS depending on the traffic-mode set for this m3ua
                        route. Possible values are:
                        1. Override
                        2. Loadshare
                        3. Broadcast (Broadcast is not yet supported by M3UA)

    <stack-name>  -   Name of the stack on which this command is executed.
                        If this is not passed, the first stack configured in
                        ShellExecutor will be used.

EXAMPLES
    m3ua route add AS1 2 -1 -1
```

Using GUI

Procedure: Create a new M3UA Route using GUI

1. Navigate to the 'Routers' tab in the M3UA Management Unit window and click on the 'Create/Remove' button.
2. This will launch the pop-up 'Create/Remove Route', where you can add values for DPC, OPC, SI, traffic-mode and Application Server Name. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new Route will be configured with parameters as specified. If there is an error in creating the Route then you will find the details of the error in the Management Console Log section below.

7.3.16. Delete a M3UA Route

Using CLI

You can delete a M3UA Route by issuing the command `m3ua route remove` with appropriate parameters as described below:

```
Name
    m3ua route remove

SYNOPSIS
    m3ua route remove <as-name> <dpc> <opc> <si> stackname <stack-name>

DESCRIPTION
    This command is used to remove a previously configured route.

PARAMETERS

    Standard Parameters

    <as-name>     -   Name of the AS assigned to route message for
                      this dpc.

    <dpc>         -   Destination Point Code.

    <opc>         -   Originating Point Code.

    <si>          -   Service Indicator.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    m3ua route remove AS1 2 -1 -1
```

Using GUI

Procedure: Delete a M3UA Route using GUI

1. Navigate to the 'Routers' tab in the M3UA Management Unit window and click on the 'Create/Remove' button.
2. This will launch the pop-up 'Create/Remove Route'. Enter the values for DPC, OPC, SI and Application Server Name that you wish to remove from the list of Routes. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Click on the 'Remove' button to delete the Route corresponding to the parameters specified.

7.4. SCCP Management

SCCP provides connectionless and connection-oriented network services. This includes address(GTT) translation and routing, flow control, segmentation and reassembly. A global title is an address (e.g., a dialed 800 number, calling card number, or mobile subscriber identification

number) which is translated by SCCP into a destination point code and subsystem number. A subsystem number uniquely identifies an application at the destination signaling point. is used as the transport layer for -based services.

The first step in configuring SCCP is defining service access points (sap). This step is mandatory. Each SCCP stack can use one or more Mtp3UserPart (Refer [_configuring_sccp] about Mtp3UserPart settings). A sap is a logical definition of the Mtp3UserPart [corresponding local SPC, network indicator (NI) and a set of destinations (remote SPC list)].

The second step is the definition of a list of available remote signaling pointcodes (SPC - rsp) and a list of available remote Sub-Systems (SNN - rss). This step is also mandatory. If routing only by GlobalTitle is used then it is not required to configure remote Sub-Systems.

Since acts as a message router, it is required to configure routing information. Rules (rule), primary and backup (address) (if backup addresses are available) addresses should be configured. If XUDT and LUDT messages are available in the SS7 network, you should configure a set of long message rules (lmr) that will allow long messages. This step is not mandatory. If no long message rules are configured only UDT messages will be used.

The last step is optional. You can configure a set of concerned signaling point codes (csp). Each point code will be announced when local SCCP user becomes unavailable.

7.4.1. Using CLI

You can manage all SCCP related configurations through the Command Line Interface by using the `sccp` command with appropriate parameters. You can create, modify, delete and view SCCP Service Access Points (sap) and Destinations (dest), Remote Signaling Point Codes (rsp), Remote Sub Systems (rss), Concerned Signaling Point Codes (csp), Routing information (rules, primary and backup addresses) and Long Message Rules (lmr). You can also set and get values for general parameters using this command.

7.4.2. Using GUI

The GUI will allow you to manage your SCCP configurations efficiently using a user-friendly interface. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'SCCP' link in the left panel. The main panel will display the names of all configured SCCP Management units. To configure or view the settings of a particular SCCP Management Unit you must click on the name of that unit. The GUI will look similar to the figure below and is divided into tabs.

The first tab will display the properties of the SCCP Management unit. These details displayed here are fetched from the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name. These properties can be modified here in the GUI. To modify them you must click on pencil, make changes and save. The GUI will then display the modified values.

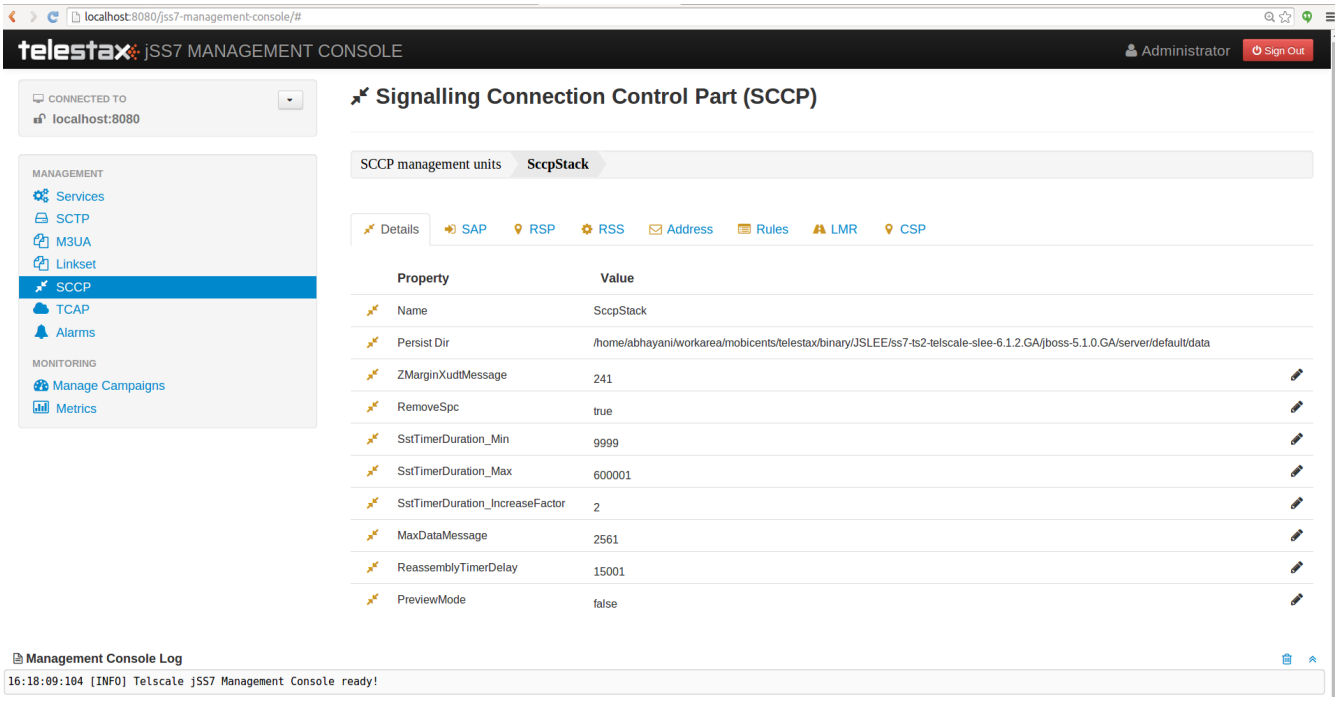


Figure 25. GUI - SCCP Management

The other seven tabs will allow you to manage all SCCP configurations within this SCCP Management unit.

7.4.3. SCCP stack properties

SCCP protocol version

Using CLI

We can specify which protocol specification will use SCCP stack (ITU-T or ANSI).

You can set the 'sccpprotocolversion' by issuing the command `sccp set sccpprotocolversion` with appropriate parameters as described below. You can verify this by issuing the command `sccp get sccpprotocolversion` which will display the value set for this property.

```
Name
    sccp set sccpprotocolversion

SYNOPSIS
    sccp set sccpprotocolversion <ITU | ANSI> stackname <stack-name>

DESCRIPTION
    Sets the value for sccpprotocolversion property ITU or ANSI.
    Default value is ITU.

PARAMETERS

    Optional Parameters

    <stack-name>    -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp set sccpprotocolversion ITU
```

Using GUI

On SCCP management page, click on pencil against the 'ReservedForNationalUseValue of AddressIndicator' property and text box becomes editable. Change value and save.

Maximum Data Message

Using CLI

You can set the 'maxdatamessage' by issuing the command `sccp set maxdatamessage` with appropriate parameters as described below. You can verify this by issuing the command `sccp get maxdatamessage` which will display the value set for this property.

```
Name
    sccp set maxdatamessage

SYNOPSIS
    sccp set maxdatamessage <maxdatamessage> stackname <stack-name>

DESCRIPTION
    Sets Max available SCCP message data for all message types. Range is 2560 to 3952.
    If passed value is less than 2560, it sets to 2560 and if passed value is greater
    than 3952, it sets to 3952.

PARAMETERS

    Standard Parameters

    <maxdatamessage>  -   Maximum data message size.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp set maxdatamessage 30000
```

Using GUI

On SCCP management page, click on pencil against the 'PreviewMode' property and text box becomes editable. Change value and save.

Preview Mode

Using CLI

You can set the 'previewmode' by issuing the command `sccp set previewmode` with appropriate parameters as described below. You can verify this by issuing the command `sccp get previewmode` which will display the value set for this property.

```
Name
    sccp set previewmode

SYNOPSIS
    sccp set previewmode <true | false> stackname <stack-name>

DESCRIPTION
    If set to true, stack only listens for incoming messages and does not
    send anything out of stack. Messages are silently dropped.

PARAMETERS

    Standard Parameters

    <previewmode> -   Set preview mode to true or false.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp set previewmode false
```

Using GUI

On SCCP management page, click on pencil against the 'PreviewMode' property and text box becomes editable. Change value and save.

Reassembly Timer Delay

Using CLI

You can set the 'reassemblytimerdelay' by issuing the command `sccp set reassemblytimerdelay` with appropriate parameters as described below. You can verify this by issuing the command `sccp get reassemblytimerdelay` which will display the value set for this property.

```
Name
    sccp set reassemblytimerdelay

SYNOPSIS
    sccp set reassemblytimerdelay <reassemblytimerdelay> stackname <stack-name>

DESCRIPTION
    Sets SCCP segmented message reassembling timeout (in milliseconds).
    Range is 10000 to 20000. If passed value is less than 10000, it sets to 10000 and
    if passed value is greater than 20000, it sets to 20000

PARAMETERS

    Standard Parameters

    <reassemblytimerdelay>    -   Re-assembly time delay in milliseconds

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp set reassemblytimerdelay 20000
```

Using GUI

On SCCP management page, click on pencil against the 'ReassemblyTimerDelay' property and text box becomes editable. Change value and save.

Remove Signaling Point Code

Using CLI

You can set the 'removespc' by issuing the command `sccp set removespc` with appropriate parameters as described below. You can verify this by issuing the command `sccp get removespc` which will display the value set for this property.

```
Name
    sccp get removespc

SYNOPSIS
    sccp get removespc stackname <stack-name>

DESCRIPTION
    Gets the value for removespc property.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp get removespc
```

Using GUI

On SCCP management page, click on pencil against the 'RemoveSpc' property and text box becomes editable. Change value and save.

SST Timer Duration Increase Factor

Using CLI

You can set the 'ssttimerduration_increasefactor' by issuing the command `sccp set ssttimerduration_increasefactor` with appropriate parameters as described below. You can verify this by issuing the command `sccp get ssttimerduration_increasefactor` which will display the value set for this property.

```
Name
    sccp get ssttimerduration_increasefactor

SYNOPSIS
    sccp get ssttimerduration_increasefactor stackname <stack-name>

DESCRIPTION
    Gets the value for ssttimerduration_increasefactor property.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp get ssttimerduration_increasefactor
```

Using GUI

On SCCP management page, click on pencil against the 'SstTimerDuration_IncreaseFactor' property and text box becomes editable. Change value and save.

SST Timer Duration Max

Using CLI

You can set the 'ssttimerduration_max' by issuing the command `sccp set ssttimerduration_max` with appropriate parameters as described below. You can verify this by issuing the command `sccp get ssttimerduration_max` which will display the value set for this property.

```
Name
    sccp get ssttimerduration_max

SYNOPSIS
    sccp get ssttimerduration_max stackname <stack-name>

DESCRIPTION
    Gets the value for ssttimerduration_max property.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp get ssttimerduration_max
```

Using GUI

On SCCP management page, click on pencil against the 'SstTimerDuration_Max' property and text box becomes editable. Change value and save.

SST Timer Duration Min

Using CLI

You can set the 'ssttimerduration_min' by issuing the command `sccp set ssttimerduration_min` with appropriate parameters as described below. You can verify this by issuing the command `sccp get ssttimerduration_min` which will display the value set for this property.

```
Name
    sccp get ssttimerduration_min

SYNOPSIS
    sccp get ssttimerduration_min stackname <stack-name>

DESCRIPTION
    Gets the value for ssttimerduration_min property.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp get ssttimerduration_min
```

Using GUI

On SCCP management page, click on pencil against the 'SstTimerDuration_Max' property and text box becomes editable. Change value and save.

ZMargin XUDT Message

Using CLI

You can set the 'zmarginxudtmessage' by issuing the command `sccp set zmarginxudtmessage` with appropriate parameters as described below. You can verify this by issuing the command `sccp get zmarginxudtmessage` which will display the value set for this property.

```
Name
    sccp get ssttimerduration_min

SYNOPSIS
    sccp get ssttimerduration_min stackname <stack-name>

DESCRIPTION
    Gets the value for ssttimerduration_min property.

PARAMETERS

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp get ssttimerduration_min
```

Using GUI

On SCCP management page, click on pencil against the 'ZMarginXudtMessage' property and text box becomes editable. Change value and save.

7.4.4. View all Service Access Points (SAP)

Using CLI

You can view the details of all configured Service Access Points by issuing the command `sccp sap show` as described below:

```
Name
    sccp sap show

SYNOPSIS
    sccp sap show <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all Service Access Points. If an <id>
    is specified, the command will only display the details of the SAP identified by
    the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>          -   The id of the SAP whose details are to be displayed. If
                      this parameter is not specified, the details of all
                      defined SAPs will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'SAP' tab. Here you can view a list of all the Service Access Points created. Every correctly configured Service Access Point will be displayed in a row with their defined values. The last column 'Action' will allow you to delete the Service Access Point.

7.4.5. Create a new Service Access Point

Using CLI

You can create a new Service Access Point by issuing the command `sccp sap create` with appropriate parameters as described below:

```
Name
    sccp sap create

SYNOPSIS
    sccp sap create <id> <mtp3-id> <opc> <ni> stackname <stack-name> networkid
<networkId>

DESCRIPTION
    This command is used to define a new Service Access Point.

PARAMETERS

    Standard Parameters

    <id>          -   The newly defined SAP will be identified using this 'id'.
                    This must be a unique number.

    <mtp3-id>     -   Mtp3UserPart index - used as an index of 'mtp3UserPart'
                    property of the SccpStack Bean. For each Mtp3UserPart,
                    a sap must be configured.

    <opc>         -   MTP Point code of the local signaling point. Takes an
                    Integer Value.

    <ni>          -   Network indicator that forms part of the
                    Service Information Octet (SIO).

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

    <networkId>     - A digital parameter that means to which virtual SS7 network
                    belongs Service Access Point (SAP). If this parameter is
                    skipped - networkId will be set to "0" when SAP creation.
                    Refer <xref linkend="design_multitenancy"/>.


EXAMPLES
    sccp sap create 1 1 101 2

    The above command will create a new Service Access Point identified by the
    number '1'. The values for 'mtp3-id', 'opc' and 'ni' are 1, 101 and 2
respectively.
```

Using GUI

Procedure: Create a new Service Access Point using GUI

1. Navigate to the 'SAP' tab in the SCCP Management window and click on the 'Create SAP' button. This will launch a pop-up 'Create SAP'.
2. In the 'Create SAP' pop-up, add details of the new Service Access Point. You must ensure that you fill in all the mandatory parameters (Id, MTP3 Id, OPC, NI). For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new SAP will be created with parameters as specified. If there is an error in creating the SAP then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create SAP' pop-up.

7.4.6. Modify a Service Access Point

Using CLI

You can modify the values of a Service Access Point by issuing the command `sccp sap modify` with appropriate parameters as described below:

```
Name
    sccp sap modify

SYNOPSIS
    sccp sap modify <id> <mtp3-id> <opc> <ni> stackname <stack-name> networkid
<networkId>

DESCRIPTION
    This command is used to modify a previously defined Service Access Point.

PARAMETERS

    Standard Parameters

    <id>          -   The id of the SAP whose values are being modified.

    <mtp3-id>     -   Mtp3UserPart index - used as an index of 'mtp3UserPart'
                    property of the SccpStack Bean. For each Mtp3UserPart, a
                    sap must be configured.

    <opc>         -   MTP Point code of the local signaling point. Takes an
                    Integer Value.

    <ni>          -   Network indicator that forms part of the Service
                    Information Octet (SIO).

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

    <networkId>     - A digital parameter that means to which virtual SS7 network
                    belongs Service Access Point (SAP). If this parameter is
                    skipped - networkId will be set to "0" when SAP creation.
                    Refer <xref linkend="design_multitenancy"/>.


EXAMPLES
    sccp sap modify 1 2 102 2

    The above command will modify the values of the Service Access Point identified
    by the number '1'. The new values for 'mtp3-id', 'opc' and 'ni' are 2, 102 and 2
    respectively.
```

7.4.7. Delete a Service Access Point

Using CLI

You can delete a SAP by issuing the command `sccp sap delete` with appropriate parameters as described below:

```
Name
    sccp sap delete

SYNOPSIS
    sccp sap delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to delete a previously defined Service Access Point.

PARAMETERS

    Standard Parameters

    <id>          -   The id of the SAP that is being deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    sccp sap delete 1

    The above command will delete the SAP identified by the number '1'.
```

Using GUI

Procedure: Delete a SAP using GUI

1. Navigate to the 'SAP' tab in the SCCP Management Unit window and locate the row corresponding to the SAP you wish to delete.
2. The last column for 'Delete' action will display a 'x' button in red and will be enabled.
3. Click on the red 'x' button to delete the corresponding SAP.

7.4.8. View all Destinations specified for a SAP

Using CLI

You can view the details of all Destinations specified for a Service Access Point by issuing the command `sccp dest show` as described below:

```
Name
    sccp dest show

SYNOPSIS
    sccp dest show <sap-id> <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all Destinations specified for a
    Service Access Point. If an <id> is specified in the command, it will only display
    the details of the Destination identified by the value of the 'id' specified.

PARAMETERS

    Standard Parameters

    <sap-id>  -   The id of the SAP whose Destination details are to be
                displayed.


    Optional Parameters

    <id>          -   The id of the Destination whose details are to be
                    displayed. If this parameter is not specified, the details
                    of all Destinations defined within the SAP 'sap-id' will
                    be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'SAP' tab. Here you can view a list of all the Service Access Points created. Every correctly configured Service Access Point will be displayed in a row with their defined values. To view the Destination details of a specific SAP, click on the row corresponding to the SAP. The row will expand below to display the details of all configured Destinations.

7.4.9. Define a new Destination for a SAP

Using CLI

You can define a new Destination for a Service Access Point by issuing the command `sccp dest create` with appropriate parameters as described below:

```
Name
    sccp dest create

SYNOPSIS
```

```
    sccp dest create <sap-id> <id> <first-dpc> <last-dpc> <first-sls> <last-sls>
    <sls-mask> stackname <stack-name>

DESCRIPTION
    This command is used to define a new Destination for a Service Access Point.
    For every SAP in the system, you should configure one or more Destinations.

PARAMETERS

    Standard Parameters

    <sap-id>  -   The identifier of the SAP for which this new Destination
                is being defined. You must ensure that the SAP has been
                created prior to issuing this command.

    <id>      -   An identifier for this newly created Destination.
                The number must be unique within each SAP.

    <first-dpc>   -   The first value of the remote signaling point codes range.

    <last-dpc>    -   The last value of the remote signaling point codes range.
                If the Destination specifies only a single Signaling
                Point Code, this value must be equal to the value
                specified for 'first-dpc'.

    <first-sls>   -   The first value of the SLS range.
                SLS value range is from 0 to 255.

    <last-sls>    -   The last value of the SLS range.

    <sls-mask>    -   The mask value. SLS of a message will be exposed by
                performing a bitwise AND operation with this mask prior to
                comparing it with first-sls and last-sls values.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.


EXAMPLES
    sccp dest create 1 1 201 201 0 7 7

    The above command will create a new Destination (identified by number '1') for
    a Service Access Point identified by the number '1'. The values for 'first-dpc',
    'last-dpc', 'first-sls', 'last-sls' and 'sls-mask' are 201, 201, 0, 7 and 7
    respectively.

    sccp dest create 2 1 300 399 0 255 255
```

```
    The above command will create a new Destination (identified by number '2') for
    a Service Access Point identified by the number '1'. The values for 'first-dpc',
    'last-dpc', 'first-sls', 'last-sls' and 'sls-mask' are 300, 399, 0, 255 and 255
    respectively. This Destination will cover all possible SLS values.
    Therefore the value for first-sls =0, last-sls=255 and sls-mask=255
```

Using GUI

Procedure: Define a new Destination for a Service Access Point using GUI

1. Navigate to the 'SAP' tab in the SCCP Management window and click on the row corresponding to the SAP for which you would like to define a new Destination.
2. The SAP row will expand below to display the details of all configured Destinations. In this section for Destinations, click on the 'Create Destination' button. This will launch a new pop-up 'Create MTP3 Destination'.
3. In the 'Create MTP3 Destination' pop-up, add details of the new Destination being defined for the Service Access Point. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
4. Verify the details entered and then click on the 'Create' button. A new Destination will be created with parameters as specified. If there is an error in creating the SAP then you will find the details of the error in the Management Console Log section below.
5. Click on the 'Close' button to close the 'Create MTP3 Destination' pop-up.

7.4.10. Modify a Destination defined for a SAP

Using CLI

You can modify the values of a Destination defined for a Service Access Point by issuing the command `sccp dest modify` with appropriate parameters as described below:

```
Name
    sccp dest modify

SYNOPSIS
    sccp dest modify <sap-id> <id> <first-dpc> <last-dpc> <first-sls> <last-sls>
    <sls-mask> stackname <stack-name>


DESCRIPTION
    This command is used to modify the values of a Destination previously defined for
    a Service Access Point.

PARAMETERS

    Standard Parameters

    <sap-id>  -   The identifier of the SAP whose Destination is being
```

```
                modified.

    <id>      -   The identifier of the Destination that is being modified.

    <first-dpc>   -   The first value of the remote signaling point codes range.

    <last-dpc>    -   The last value of the remote signaling point codes range.
                If the Destination specifies only a single Signaling
                Point Code, this value must be equal to the value
                specified for 'first-dpc'.

    <first-sls>   -   The first value of the SLS range.
                SLS value range is from 0 to 255.

    <last-sls>    -   The last value of the SLS range.

    <sls-mask>    -   The mask value. SLS of a message will be exposed by
                performing a bitwise AND operation with this mask prior to
                comparing it with first-sls and last-sls values.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp dest modify 1 1 201 299 0 255 255

    The above command will modify the values of the Destination identified by the
    number '1' within the Service Access Point identified by the number '1'.
    The new values for 'first-dpc', 'last-dpc', 'first-sls', 'last-sls' and 'sls-mask'
    are 201, 299, 0, 255 and 255 respectively.
```

7.4.11. Delete a Destination defined for a SAP

Using CLI

You can delete a Destination defined for a SAP by issuing the command `sccp dest delete` with appropriate parameters as described below:

```
Name
    sccp dest delete

SYNOPSIS
    sccp dest delete <sap-id> <id> stackname <stack-name>

DESCRIPTION
    This command is used to remove a previously defined Destination from a
    Service Access Point.

PARAMETERS

    Standard Parameters

    <sap-id>  -   The identifier of the SAP whose Destination is being
                deleted.

    <id>      -   The identifier of the Destination that is being
                deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp dest delete 1 1

    The above command will delete the Destination identified by the number '1' from
    the SAP identified by the number '1'.
```

Using GUI

Procedure: Delete a Destination defined for a SAP using GUI

1. Navigate to the 'SAP' tab in the SCCP Management Unit window and click on the row corresponding to the SAP from which you wish to delete a Destination.
2. The SAP row will expand below to display the details of all configured Destinations. In this section for Destinations, locate the specific Destination you wish to remove from the list.
3. In the row corresponding to the identified Destination, click on the red 'x' button in the actions column to delete that Destination.

7.4.12. View all configured SCCP Addresses

Using CLI

You can view the details of all configured SCCP Addresses by issuing the command `sccp address show` as described below:

```
Name
    sccp address show

SYNOPSIS
    sccp address show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured addresses.
    If an <id> is specified in the command, it will only display the details of the
    Address identified by the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>          -   The id of the Address whose details are to be displayed.
                    If this parameter is not specified, the details of all
                    configured Addresses will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'Address' tab. Here you can view a list of all the configured Addresses. Every correctly configured Address will be displayed in a row along with the defined values.

7.4.13. Create a new Primary/Backup address

Using CLI

You can create a new primary address or backup address of translation by issuing the command `sccp address create` with appropriate parameters as described below:

```
Name
    sccp address create

SYNOPSIS
    sccp address create <id> <address-indicator> <point-code> <subsystem-number>
    <translation-type> <numbering-plan> <nature-of-address-indicator> <digits>
    stackname <stack-name>

DESCRIPTION
    This command is used to create a new primary address or backup address of
    translation. You can create a new newCallingParty address as well
    using this command. The global title address information of this command is
combined with
```

```
    the global title being translated by examining the mask provided in the
    'sccp rule create' command.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this address.

    <address-indicator>   -   The address indicator is the first field in a SCCP
                    Party Address (called/calling) and is one octet in
                    length. Its function is to indicate which
                    information elements are present so that the
                    address can be interpreted. In other words, it
                    indicates the type of addressing information that
                    is to be found in the address field. The
                    addressing information from the original global
                    title is then compared with the passed address
                    information to match the rule.

                    SCCP ADDRESS INDICATOR
                    -------------------------------
                    | 8 | 7 | 6 | 5 | 4 |3 | 2 | 1 |
                    -------------------------------

                    Bit '1'     : PC Indicator
                              (1 = included)

                    Bit '2'     : SSN Indicator
                              (1 = included)

                    Bit '3 - 6' : GT Indicator
                              (0000 = GT not included)
                              (0001 = GT includes
                                  Nature of Address)
                              (0010 = GT includes
                                  Translation Type)
                              (0011 = GT includes
                                  Translation Type,
                                  Numbering Plan and
                                  Encoding Scheme)
                              (0100 = GT includes
                                  Translation Type,
                                  Numbering Plan and
                                  Encoding Scheme and
                                  Nature of Address)

                    Bit '7'     : Routing Indicator
                              (0 = Route on GT,
                              1 = Route on PC + SSN)
```

```
                Bit '8'     : Reserved for National use.

                Only two fields of Address Indicator is used now for GTT:
                - GT Indicator (this is used for GlobalTitle type that
                  will be created)
                - Routing Indicator bit (0 = Route on GT, 1 = Route on
                  PC + SSN)
                GT Indicator for ITU-T network that is mostly used is -
                0100 (GT includes Translation Type, Numbering Plan and
                Encoding Scheme and Nature of Address). Digital value
                for it is - 16.
                For 0100 GT Indicator we will use two possible values:
                16 - 0100 GT Indicator and Route on GT
                80 - 0100 GT Indicator and Route on PC + SSN
                Even when SCCP stack works in ANSI mode Address Indicator
                value for CLI and GUI must have values that we use for
                ITU-T mode.

<point-code>      -   MTP Signaling Point Code.
                This parameter contains a point code to which message will
                be routed after GTT (DPC field).
                This parameter is mandatory.

<subsystem-number>    - This parameter contains SSN which will be placed
                into CalledPartyAddress. If you set this parameter to "0",
                SSN from CalledPartyAddress of an original message will be
                put into CalledPartyAddress.

<translation-type>    -   This is ignored if GT Indicator is 0000 or 0001.

                TRANSLATION TYPE VALUES

                Value       Description
                ---------------------------------
                0       Unknown

                1  - 63     International Service

                64 - 127    Spare

                128 - 254   National Network Specific

                255     Reserved for Expansion

                Value of this parameter will be placed into
                CalledPartyAddress. This parameter is mandatory if GT
                Indicator suppose this parameter is included into GT.
                Most used value: 0 - translation-type - Unknown

<numbering-plan>  -   The Number Plan (NP) field specifies the numbering
                plan which the address information follows. This
```

```
                    is ignored if GT Indicator is 0000, 0001 or 0010.

                    Value of this parameter will be placed into
                    CalledPartyAddress. This parameter is mandatory if GT
                    Indicator suppose this parameter is included into GT.
                    Most used value: 1 - numbering-plan - ISDN/telephony

    <nature-of-address>   -   The Nature of Address Indicator (NAI) field
                    defines the address range for a specific numbering
                    plan. This is only used if GT Indicator is 0100.

                    Value of this parameter will be placed into
                    CalledPartyAddress. This parameter is mandatory if GT
                    Indicator suppose this parameter is included into GT.
                    Most used value: 4 - nature-of-address - International

    <digits>      -   The global title address information to translate
                    to. Specified as string of digits divided into
                    subsections using separator '/' depending on if
                    the mask contains separator or not.
                    The digits string can contain:

                    DIGIT PATTERN

                    Value       Description
                    ---------------------------------
                    -       padding - ignored

                    /       separator used to split the digit
                            pattern into sections.
                            Each section is processed
                            differently as specified by the
                            mask parameter in the
                            'sccp rule create' command.

                    We need this parameter if at least one section of Rule
                    mask contains "R" (replace) value. Else set this field
                    to "0". If this field is needed it should contain the
                    same subsections as the rule mask has.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp address create 1 71 2 8 0 0 3 123456789
```

Using GUI

Procedure: Create a new Primary/Backup Address using GUI

1. Navigate to the 'Address' tab in the SCCP Management window and click on the 'Create Address' button. This will launch a new pop-up 'Create SCCP Address'.
2. In the 'Create SCCP Address' pop-up, add details of the new SCCP Address being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new Address will be created with parameters as specified. If there is an error in creating the Address then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create SCCP Address' pop-up.

7.4.14. Modify a Primary/Backup Address

Using CLI

You can modify the values of a primary address or backup address of translation by issuing the command `sccp address modify` with appropriate parameters as described below:

```
Name
    sccp address modify

SYNOPSIS
    sccp address modify <id> <address-indicator> <point-code> <subsystem-number>
    <translation-type> <numbering-plan> <nature-of-address-indicator> <digits>
    stackname <stack-name>

DESCRIPTION
    This command is used to modify the values of an address
    previously defined.

PARAMETERS

    Standard Parameters

    <id>          -   Identifier of the address to be modified.

    <address-indicator>   -   The address indicator is the first field in a SCCP
                    Party Address (called/calling) and is one octet in
                    length. Its function is to indicate which
                    information elements are present so that the
                    address can be interpreted. In other words, it
                    indicates the type of addressing information that
                    is to be found in the address field. The
                    addressing information from the original global
                    title is then compared with the passed address
                    information to match the rule.
```

```
                SCCP ADDRESS INDICATOR
                --------------------------------
                | 8 | 7 | 6 | 5 | 4 |3 | 2 | 1 |
                --------------------------------

                Bit '1'     : PC Indicator
                          (1 = included)

                Bit '2'     : SSN Indicator
                          (1 = included)

                Bit '3 - 6' : GT Indicator
                          (0000 = GT not included)
                          (0001 = GT includes
                              Nature of Address)
                          (0010 = GT includes
                              Translation Type)
                          (0011 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme)
                          (0100 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme and
                              Nature of Address)

                Bit '7'     : Routing Indicator
                          (0 = Route on GT,
                          1 = Route on PC + SSN)

                Bit '8'     : Reserved for National use.

                Only two fields of Address Indicator is used now for GTT:
                - GT Indicator (this is used for GlobalTitle type that
                  will be created)
                - Routing Indicator bit (0 = Route on GT, 1 = Route on
                  PC + SSN)
                GT Indicator for ITU-T network that is mostly used is -
                0100 (GT includes Translation Type, Numbering Plan and
                Encoding Scheme and Nature of Address). Digital value
                for it is - 16.
                For 0100 GT Indicator we will use two possible values:
                16 - 0100 GT Indicator and Route on GT
                80 - 0100 GT Indicator and Route on PC + SSN
                Even when SCCP stack works in ANSI mode Address Indicator
                value for CLI and GUI must have values that we use for
                ITU-T mode.

    <point-code>        -   MTP Signaling Point Code.
```

```
                This parameter contains a point code to which message will
                be routed after GTT (DPC field).
                This parameter is mandatory.

<subsystem-number>  - This parameter contains SSN which will be placed
                into CalledPartyAddress. If you set this parameter to "0",
                SSN from CalledPartyAddress of an original message will be
                put into CalledPartyAddress.

<translation-type>    -   This is ignored if GT Indicator is 0000 or 0001.

                TRANSLATION TYPE VALUES

                Value       Description
                ---------------------------------
                0       Unknown

                1  - 63     International Service

                64 - 127    Spare

                128 - 254   National Network Specific

                255     Reserved for Expansion

                Value of this parameter will be placed into
                CalledPartyAddress. This parameter is mandatory if GT
                Indicator suppose this parameter is included into GT.
                Most used value: 0 - translation-type - Unknown

<numbering-plan>    -   The Number Plan (NP) field specifies the numbering
                plan which the address information follows. This
                is ignored if GT Indicator is 0000, 0001 or 0010.

                Value of this parameter will be placed into
                CalledPartyAddress. This parameter is mandatory if GT
                Indicator suppose this parameter is included into GT.
                Most used value: 1 - numbering-plan - ISDN/telephony

<nature-of-address> -   The Nature of Address Indicator (NAI) field
                defines the address range for a specific numbering
                plan. This is only used if GT Indicator is 0100.

                Value of this parameter will be placed into
                CalledPartyAddress. This parameter is mandatory if GT
                Indicator suppose this parameter is included into GT.
                Most used value: 4 - nature-of-address - International

<digits>      -   The global title address information to translate
                to. Specified as string of digits divided into
                subsections using separator '/' depending on if
```

```
                    the mask contains separator or not.
                    The digits string can contain:

                    DIGIT PATTERN

                    Value       Description
                    ---------------------------------
                    -       padding - ignored

                    /       separator used to split the digit
                            pattern into sections.
                            Each section is processed
                            differently as specified by the
                            mask parameter in the
                            'sccp rule create' command.

                    We need this parameter if at least one section of Rule
                    mask contains "R" (replace) value. Else set this field
                    to "0". If this field is needed it should contain the
                    same subsections as the rule mask has.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

7.4.15. Delete a Primary/Backup Address

Using CLI

You can delete a Primary or Backup Address by issuing the command `sccp address delete` with appropriate parameters as described below:

```
Name
    sccp address delete

SYNOPSIS
    sccp address delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to remove previously defined addresses.

PARAMETERS

    Standard Parameters

    <id>      -   The identifier of the address that is being deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp address delete 1

    The above command will delete the address identified by the number '1'.
```

Using GUI

Procedure: Delete a Primary/Backup Address using GUI

1. Navigate to the 'Address' tab in the SCCP Management Unit window and locate the row corresponding to the Address you wish to delete.
2. In the row corresponding to the identified Address, click on the red 'x' button in the actions column to delete that Address.

7.4.16. View all configured SCCP Rules

Using CLI

You can view the details of all configured SCCP Rules by issuing the command `sccp rule show` as described below:

```
Name
    sccp rule show

SYNOPSIS
    sccp rule show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all Rules configured. If an <id> is
    specified in the command, it will only display the details of the Rule identified
    by the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>      -   The id of the Rule whose details are to be displayed.
                  If this parameter is not specified, the details of all
                  configured Rules will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                  If not passed, the first stack configured in ShellExecutor
                  will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'Rules' tab. Here you can view a list of all the configured Rules. Every correctly configured Rule will be displayed in a row along with the defined values.

7.4.17. Create a new SCCP Rule

Sorting Algorithm

When you define Rules, a comparison function imposes ordering the collection of SCCP Rules using a sorting algorithm that is based on the GT digits. The algorithm is defined below:

1. Rules defined with OriginationType==localOriginated or OriginationType==remoteOriginated, are always at the top of the list. Rules defined with OriginationType==All are always at the bottom of the list.

   Among rules with the same values for OriginationType (All or localOriginated/remoteOriginated), the sorting is done using the below rules.

2. Rules with GT digits having no wildcard (* or ?) are always at the top of the list. Between two Rules with GT digits, both having no wildcards, the one with the shortest length is at the top of the list. For example, Digit1 "123456" will be above Digit2 "1234567890" and this will be above Digit3 "999/*
3. Rules with GT digits having the wildcard "?" are always above digits having the wildcard "*". For example, Digit1 "800/????/9" will be above Digit2 "999/*

4. Between Rules with two GT digits both having wildcard "?", the one with the least number of wildcard "?" is at the top of the list. For example, Digit1 "800/????/9" will be above Digit2 "800/?????/9"
5. Between Rules with two GT digits both having an equal number of wildcard "?", the digit whose first appearance of "?" is after other, is at the top of the list. For example between Digit1 "80/??/0/???/9" and Digit 2 "800/?????/9", Digit2 is above Digit1.

When a Rule is compared during Translation, comparison always starts from the top of the list.

Using CLI

You can create a new Rule by issuing the command `sccp rule create` with appropriate parameters as described below:

```
Name
    sccp rule create

SYNOPSIS
    sccp rule create <id> <mask> <address-indicator> <point-code> <subsystem-number>
    <translation-type> <numbering-plan> <nature-of-address-indicator> <digits>
    <ruleType> <primary-address-id> backup-addressid <backup-address-id>
    loadsharing-algo <loadsharing-algorithm> newcgparty-addressid
    <new-callingPartyAddress-id> origination-type <originationType>
    stackname <stack-name> networkid <networkId>

DESCRIPTION
    This command is used to create a new SCCP Routing Rule. You must ensure that
    primary and backup addresses are configured properly prior to executing this
    command.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this Rule.

    <mask>            -   A mask defines which part of the originally dialed
                    digits remains in the translated digits and which
                    part is replaced by the digits from primary or
                    backup address. A mask is divided into sections by
                    separator '/'. The number of sections in a mask
                    should be equal to the sections in digits passed
                    in this command and the sections in primary or
                    backup address. This paramter is mandatory.

                    MASK DEFINITIONS

                    Mnemonic    Function
                    ---------------------------------
                    -       Ignore
```

```
                /       Separator used to split the mask
                        into sections.

                K       Retain the original dialed digits
                        of this section in the translated
                        digits.

                R       Replace the original dialed digits
                        of this section with the same
                        section from primary or backup
                        address in the translated digits.

<address-indicator>   -   The address indicator is the first field in a SCCP
                Party Address (called/calling) and is one octet in
                length. Its function is to indicate which
                information elements are present so that the
                address can be interpreted. In other words, it
                indicates the type of addressing information that
                is to be found in the address field. The
                addressing information from the original global
                title is then compared with the passed address
                information to match the rule.

                SCCP ADDRESS INDICATOR
                --------------------------------
                | 8 | 7 | 6 | 5 | 4 |3 | 2 | 1 |
                --------------------------------

                Bit '1'     : PC Indicator
                          (1 = included)

                Bit '2'     : SSN Indicator
                          (1 = included)

                Bit '3 - 6' : GT Indicator
                          (0000 = GT not included)
                          (0001 = GT includes
                              Nature of Address)
                          (0010 = GT includes
                              Translation Type)
                          (0011 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme)
                          (0100 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme and
                              Nature of Address)
```

```
                Bit '7'     : Routing Indicator
                          (0 = Route on GT,
                          1 = Route on PC + SSN)

                Bit '8'     : Reserved for National use.

                Only GT Indicator is used in the current implementation.
                A Rule matches to an original address only if GT Indicator
                from address-indicator is the same in a rule and in an
                original address. GT Indicator for ITU-T network that is
                mostly used is - 0100 (GT includes Translation Type,
                Numbering Plan and Encoding Scheme and Nature of Address).
                If you also use 0100 - use digital value for it - 16.
                Even when SCCP stack works in ANSI mode Address Indicator
                value for CLI and GUI must have values that we use for
                ITU-T mode.

<point-code>      -   MTP Signaling Point Code. This is ignored if
                Bit '0' of address-indicator is not set.
                This parameter is not used in currentimplemetation and
                can be set to "0".

<subsystem-number>    -   This is ignored if Bit '1' of address-indicator is
                not set.
                This parameter is not used in currentimplemetation and
                can be set to "0".

<translation-type>    -   This is ignored if GT Indicator is 0000 or 0001.

                TRANSLATION TYPE VALUES

                Value       Description
                ---------------------------------
                0       Unknown

                1  - 63     International Service

                64 - 127    Spare

                128 - 254   National Network Specific

                255     Reserved for Expansion

                This paramter is mandatory.
                A Rule matches to an original address only if a value of
                this parameter is the same in a rule and in an original
                address. Values are compared only if GT type contains this
                parameter (see GT Indicator description in
                <address-indicator> parameter).
                Most used values:
                0 - translation-type - Unknown
```

```
<numbering-plan>  -   The Number Plan (NP) field specifies the numbering
                plan which the address information follows. This
                is ignored if GT Indicator is 0000, 0001 or 0010.

                NUMBER PLAN VALUES

                Value       Description
                ---------------------------------
                0       Unknown

                1       ISDN/Telephony Number Plan
                        (Recommendations E.163 and E.164)

                2       Generic Numbering Plan

                3       Data Numbering Plan
                        (Recommendations X.121)

                4       Telex Numbering Plan
                        (Recommendations F.69)

                5       Maritime Mobile Numbering Plan
                        (Recommendations E.210, E.211)

                6       Land Mobile Numbering Plan
                        (Recommendations E.212)

                7       ISDN/Mobile Numbering Plan
                        (Recommendations E.214)

                8 to 13     Spare

                14      Private Network or
                        Network-Specific Numbering Plan

                15      Reserved

                This paramter is mandatory.
                A Rule matches to an original address only if a value of
                this parameter is the same in a rule and in an original
                address. Values are compared only if GT type contains this
                parameter (see GT Indicator description in
                <address-indicator> parameter).
                Most used values:
                1 - numbering-plan - ISDN/telephony

<nature-of-address>   -   The Nature of Address Indicator (NAI) field
                defines the address range for a specific numbering
                plan. This is only used if GT Indicator is 0100.
```

```
                  NAI VALUES

                  Value       Description
                  ---------------------------------
                  0       Unknown

                  1       Subscriber Number

                  2       Reserved for National use

                  3       National Significant Number

                  4       International Number

                  5 to 127    Spare

                  This paramter is mandatory.
                  A Rule matches to an original address only if a value of
                  this parameter is the same in a rule and in an original
                  address. Values are compared only if GT type contains this
                  parameter (see GT Indicator description in
                  <address-indicator> parameter).
                  Most used values:
                  4 - nature-of-address - International

<digits>      -   Specifies the string of digits divided into
                  subsections using separator '/' depending on if
                  the mask contains separator or not. The dialed
                  digits should match with theses digits as per the
                  rule specified below:

                  DIGIT PATTERN

                  Value       Description
                  ---------------------------------
                  -       padding - ignored

                  *       wildcard - matches any number of
                          digits

                  ?       wildcard - matches exactly one
                          digit

                  /       separator used to split the digit
                          pattern into sections.
                          Each section can be processed
                          differently as specified
                          by the mask parameter.

                  This parameter is mandatory. It should contain the same
                  subsections count as the rule mask has.
```

```
<ruleType>        -   Takes one of the following values defined below.

                RULE TYPE VALUES

                Value       Description
                ---------------------------------
                solitary    Only one (primary) address is used
                        for routing.
                        (<backup-address-id> may be missed
                        in this case).

                dominant    Both primary and backup
                        addresses are used and mandatory.
                        If both the addresses are
                        available, the primary address is
                        used for routing.

                loadshared  Both primary and backup
                        addresses are used and mandatory.
                        If both the addresses are
                        available, either primary or
                        backup address is used for
                        routing.
                        The <loadsharing-algorithm> should
                        be configured in this case.

                broadcast   Both primary and backup addresses
                        are used and are mandatory. All
                        messages are routed to both
                        addresses.

<primary-address-id>  -   Identifies the SCCP Address used as the primary
                translation.

Optional Parameters

<backup-address-id>   -   Identifies the SCCP Address used as the backup
                translation in case the pointcode specified by the
                primary address is not available. Backup address
                is used if <ruleType> is not "solitary".

<loadsharing-algorithm>   -   This parameter is mandatory if <ruleType> is
                "loadshared". The Loadsharing algorithm is
                configured here. Possible values of the parameter
                are:

                LOAD SHARING ALGORITHM VALUES

                Value       Description
                ---------------------------------
```

```
                bit4        if( (SLS & 0x10) == 0 )
                        <route to primary> else
                        <route to backup>

                        This algorithm is the best if all
                        traffic is local (mobicents stack)
                        originated

                bit3        if( (SLS & 0x08) == 0 )
                        <route to primary> else
                        <route to backup>
                        This algorithm can be used if not
                        all traffic is local
                        (mobicents stack) originated.
                        But only 8 links are acceptable in
                        both linksets.

                bit2        if( (SLS & 0x04) == 0 )
                        <route to primary> else
                        <route to backup>
                        This algorithm can be used if not
                        all traffic is local
                        (mobicents stack) originated.
                        But only 8 links are acceptable in
                        both linksets.

                bit1        if( (SLS & 0x02) == 0 )
                        <route to primary> else
                        <route to backup>
                        This algorithm can be used if not
                        all traffic is local
                        (mobicents stack) originated.
                        But only 8 links are acceptable in
                        both linksets.

                bit0        if( (SLS & 0x01) == 0 )
                        <route to primary> else
                        <route to backup>
                        This algorithm can be used if not
                        all traffic is local
                        (mobicents stack) originated.
                        But only 8 links are acceptable in
                        both linksets.


    <new-callingPartyAddress-id>
                -   This address will replace the
                    callingPartyAddresses of messages that fit a Rule.

    <originationType> -   Takes one of the following values defined below.
                    If the parameter is not defined, rule applies to
```

```
                    all messages regardless of their origination.

                    ORIGINATION TYPE VALUES

                    Value           Description
                    -----------------------------------
                    localOriginated     If this parameter is
                                "localOriginated", then a
                                rule applies only for
                                messages originating from
                                local SCCP users
                                (for example a local
                                TCAP stack).

                    remoteOriginated    If this parameter is
                                "remoteOriginated", then a
                                rule applies only for
                                messages originating from
                                SS7 network and not for
                                messages originating from
                                local SCCP users.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

    <networkId>     - A digital parameter that means to which virtual SS7 network
                    belongs a Rule. If this parameter is skipped - networkId will
                    be set to "0" when a Rule creation.
                    Refer <xref linkend="design_multitenancy"/>.

EXAMPLES
    sccp rule create 1 R 71 2 8 0 0 3 123456789 solitary 1

    sccp rule create 2 R 71 2 8 0 0 3 123456789 dominant 1 1

    sccp rule create 2 R 71 2 8 0 0 3 123456789 loadshared 1 1 bit4
```

Using GUI

Procedure: Create a new Rule using GUI

1. Navigate to the 'Rules' tab in the SCCP Management window and click on the 'Create Rule' button. This will launch a new pop-up 'Create Rule'.
2. In the 'Create Rule' pop-up, add details of the new SCCP Rule being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new Rule will be created with parameters as specified. If there is an error in creating the Rule then you will find the details of the error in the Management Console Log section below.

4. Click on the 'Close' button to close the 'Create Rule' pop-up.

7.4.18. Modify a SCCP Rule

Using CLI

You can modify the values of a Rule by issuing the command `sccp rule modify` with appropriate parameters as described below:

```
Name
    sccp rule modify

SYNOPSIS
    sccp rule modify <id> <mask> <address-indicator> <point-code> <subsystem-number>
    <translation-type> <numbering-plan> <nature-of-address-indicator> <digits>
    <ruleType> <primary-address-id> backup-addressid <backup-address-id>
    loadsharing-algo <loadsharing-algorithm> newcgparty-addressid
    <new-callingPartyAddress-id> origination-type <originationType>
    stackname <stack-name> networkid <networkId>

DESCRIPTION
    This command is used to modify the values of a SCCP Route previously defined.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this Rule.

    <mask>            -   A mask defines which part of the originally dialed
                      digits remains in the translated digits and which
                      part is replaced by the digits from primary or
                      backup address. A mask is divided into sections by
                      separator '/'. The number of sections in a mask
                      should be equal to the sections in digits passed
                      in this command and the sections in primary or
                      backup address. This paramter is mandatory.

                      MASK DEFINITIONS

                      Mnemonic    Function
                      ---------------------------------
                      -       Ignore

                      /       Separator used to split the mask
                              into sections.

                      K       Retain the original dialed digits
                              of this section in the translated
                              digits.
```

```
                R       Replace the original dialed digits
                        of this section with the same
                        section from primary or backup
                        address in the translated digits.

<address-indicator>   -   The address indicator is the first field in a SCCP
                Party Address (called/calling) and is one octet in
                length. Its function is to indicate which
                information elements are present so that the
                address can be interpreted. In other words, it
                indicates the type of addressing information that
                is to be found in the address field. The
                addressing information from the original global
                title is then compared with the passed address
                information to match the rule.

                SCCP ADDRESS INDICATOR
                --------------------------------
                | 8 | 7 | 6 | 5 | 4 |3 | 2 | 1 |
                --------------------------------

                Bit '1'     : PC Indicator
                          (1 = included)

                Bit '2'     : SSN Indicator
                          (1 = included)

                Bit '3 - 6' : GT Indicator
                          (0000 = GT not included)
                          (0001 = GT includes
                              Nature of Address)
                          (0010 = GT includes
                              Translation Type)
                          (0011 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme)
                          (0100 = GT includes
                              Translation Type,
                              Numbering Plan and
                              Encoding Scheme and
                              Nature of Address)

                Bit '7'     : Routing Indicator
                          (0 = Route on GT,
                          1 = Route on PC + SSN)

                Bit '8'     : Reserved for National use.

                Only GT Indicator is used in the current implementation.
```

```
                A Rule matches to an original address only if GT Indicator
                from address-indicator is the same in a rule and in an
                original address. GT Indicator for ITU-T network that is
                mostly used is - 0100 (GT includes Translation Type,
                Numbering Plan and Encoding Scheme and Nature of Address).
                If you also use 0100 - use digital value for it - 16.
                Even when SCCP stack works in ANSI mode Address Indicator
                value for CLI and GUI must have values that we use for
                ITU-T mode.

<point-code>      -   MTP Signaling Point Code. This is ignored if
                Bit '0' of address-indicator is not set.
                This parameter is not used in currentimplemetation and
                can be set to "0".

<subsystem-number>    -   This is ignored if Bit '1' of address-indicator is
                not set.
                This parameter is not used in currentimplemetation and
                can be set to "0".

<translation-type>    -   This is ignored if GT Indicator is 0000 or 0001.

                TRANSLATION TYPE VALUES

                Value       Description
                ---------------------------------
                0       Unknown

                1  - 63     International Service

                64 - 127    Spare

                128 - 254   National Network Specific

                255     Reserved for Expansion

                This paramter is mandatory.
                A Rule matches to an original address only if a value of
                this parameter is the same in a rule and in an original
                address. Values are compared only if GT type contains this
                parameter (see GT Indicator description in
                <address-indicator> parameter).
                Most used values:
                0 - translation-type - Unknown

<numbering-plan>  -   The Number Plan (NP) field specifies the numbering
                plan which the address information follows. This
                is ignored if GT Indicator is 0000, 0001 or 0010.

                NUMBER PLAN VALUES
```

```
                  Value       Description
                  ---------------------------------
                  0       Unknown

                  1       ISDN/Telephony Number Plan
                          (Recommendations E.163 and E.164)

                  2       Generic Numbering Plan

                  3       Data Numbering Plan
                          (Recommendations X.121)

                  4       Telex Numbering Plan
                          (Recommendations F.69)

                  5       Maritime Mobile Numbering Plan
                          (Recommendations E.210, E.211)

                  6       Land Mobile Numbering Plan
                          (Recommendations E.212)

                  7       ISDN/Mobile Numbering Plan
                          (Recommendations E.214)

                  8 to 13     Spare

                  14      Private Network or
                          Network-Specific Numbering Plan

                  15      Reserved

                  This paramter is mandatory.
                  A Rule matches to an original address only if a value of
                  this parameter is the same in a rule and in an original
                  address. Values are compared only if GT type contains this
                  parameter (see GT Indicator description in
                  <address-indicator> parameter).
                  Most used values:
                  1 - numbering-plan - ISDN/telephony

<nature-of-address>   -   The Nature of Address Indicator (NAI) field
                  defines the address range for a specific numbering
                  plan. This is only used if GT Indicator is 0100.

                  NAI VALUES

                  Value       Description
                  ---------------------------------
                  0       Unknown

                  1       Subscriber Number
```

```
                2       Reserved for National use

                3       National Significant Number

                4       International Number

                5 to 127    Spare

                This paramter is mandatory.
                A Rule matches to an original address only if a value of
                this parameter is the same in a rule and in an original
                address. Values are compared only if GT type contains this
                parameter (see GT Indicator description in
                <address-indicator> parameter).
                Most used values:
                4 - nature-of-address - International

<digits>      -   Specifies the string of digits divided into
                subsections using separator '/' depending on if
                the mask contains separator or not. The dialed
                digits should match with theses digits as per the
                rule specified below:

                DIGIT PATTERN

                Value       Description
                ---------------------------------
                -       padding - ignored

                *       wildcard - matches any number of
                        digits

                ?       wildcard - matches exactly one
                        digit

                /       separator used to split the digit
                        pattern into sections.
                        Each section can be processed
                        differently as specified
                        by the mask parameter.

                This parameter is mandatory. It should contain the same
                subsections count as the rule mask has.

<ruleType>        -   Takes one of the following values defined below.

                RULE TYPE VALUES

                Value       Description
                ---------------------------------
```

```
                solitary    Only one (primary) address is used
                        for routing.
                        (<backup-address-id> may be missed
                        in this case).

                dominant    Both primary and backup
                        addresses are used and mandatory.
                        If both the addresses are
                        available, the primary address is
                        used for routing.

                loadshared  Both primary and backup
                        addresses are used and mandatory.
                        If both the addresses are
                        available, either primary or
                        backup address is used for
                        routing.
                        The <loadsharing-algorithm> should
                        be configured in this case.

                broadcast   Both primary and backup addresses
                        are used and are mandatory. All
                        messages are routed to both
                        addresses.

<primary-address-id>  -   Identifies the SCCP Address used as the primary
                translation.

Optional Parameters

<backup-address-id>   -   Identifies the SCCP Address used as the backup
                translation in case the pointcode specified by the
                primary address is not available. Backup address
                is used if <ruleType> is not "solitary".

<loadsharing-algorithm>   -   This parameter is mandatory if <ruleType> is
                "loadshared". The Loadsharing algorithm is
                configured here. Possible values of the parameter
                are:

                LOAD SHARING ALGORITHM VALUES

                Value       Description
                ---------------------------------
                bit4        if( (SLS & 0x10) == 0 )
                        <route to primary> else
                        <route to backup>

                        This algorithm is the best if all
                        traffic is local (mobicents stack)
                        originated
```

```
                bit3        if( (SLS & 0x08) == 0 )
                        <route to primary> else
                        <route to backup>
                        This algorithm can be used if not
                        all traffic is local
                        (mobicents stack) originated.
                        But only 8 links are acceptable in
                        both linksets.

    <new-callingPartyAddress-id>
                -   This address will replace the
                    callingPartyAddresses of messages that fit a Rule.

    <originationType> -   Takes one of the following values defined below.
                    If the parameter is not defined, rule applies to
                    all messages regardless of their origination.

                    ORIGINATION TYPE VALUES

                    Value           Description
                    ----------------------------------
                    localOriginated     If this parameter is
                                "localOriginated", then a
                                rule applies only for
                                messages originating from
                                local SCCP users
                                (for example a local
                                TCAP stack).

                    remoteOriginated    If this parameter is
                                "remoteOriginated", then a
                                rule applies only for
                                messages originating from
                                SS7 network and not for
                                messages originating from
                                local SCCP users.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

    <networkId>     - A digital parameter that means to which virtual SS7 network
                    belongs a Rule. If this parameter is skipped - networkId will
                    be set to "0" when a Rule creation.
                    Refer <xref linkend="design_multitenancy"/>.
```

7.4.19. Delete a Rule

Using CLI

You can delete a Rule by issuing the command `sccp rule delete` with appropriate parameters as described below:

```
Name
    sccp rule delete

SYNOPSIS
    sccp rule delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to remove a previously defined Rule.

PARAMETERS

    Standard Parameters

    <id>          -   The identifier of the Rule that is being deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp rule delete 1

    The above command will delete the Rule identified by the number '1'.
```

Using GUI

Procedure: Delete a Rule using GUI

1. Navigate to the 'Rules' tab in the SCCP Management Unit window and locate the row corresponding to the Rule you wish to delete.
2. In the row corresponding to the identified Rule, click on the red 'x' button in the actions column to delete that Rule.

7.4.20. View all configured Remote Signaling Points (RSP)

Using CLI

You can view the details of all configured Remote Signaling Points by issuing the command `sccp rsp show` as described below:

```
Name
    sccp rsp show

SYNOPSIS
    sccp rsp show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Remote Signaling
Points.

    If an <id> is specified in the command, it will only display the details of the
    Remote Signaling Point identified by the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>      -   The id of the Remote Signaling Point whose details are
                  to be displayed. If this parameter is not specified, the
                  details of all configured Remote Signaling Points will be
                  displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'RSP' tab. Here you can view a list of all the configured Remote Signaling Pointcodes. Every correctly configured RSP will be displayed in a row along with the defined values.

7.4.21. Create a new Remote Signaling Pointcode

Using CLI

You can create a new RSP by issuing the command `sccp rsp create` with appropriate parameters as described below:

```
Name
    sccp rsp create

SYNOPSIS
    sccp rsp create <id> <remote-spc> <rspc-flag> <mask>
    stackname <stack-name>

DESCRIPTION
    This command is used to define a new Remote Signaling Point. Each remote signaling
    point that SCCP can communicate with must be configured using this command.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this
                      Remote Signaling Point.

    <remote-spc>      -   The Remote Signaling Point

    <rspc-flag>       -   32 bit value. Not used for now.
                      Reserved for future

    <mask>            -   32 bit value. Not used for now.
                      Reserved for future

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    sccp rsp create 1 6477 0 0
```

Using GUI

Procedure: Create a new RSP using GUI

1. Navigate to the 'RSP' tab in the SCCP Management window and click on the 'Create RSP' button. This will launch a new pop-up 'Create Remote Signaling Pointcode'.
2. In the 'Create Remote Signaling Pointcode' pop-up, add details of the new RSP being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new RSP will be created with parameters as specified. If there is an error in creating the RSP then you will find the details of the error in the Management Console Log section below.

4. Click on the 'Close' button to close the 'Create Remote Signaling Pointcode' pop-up.

7.4.22. Modify a Remote Signaling Pointcode

Using CLI

You can modify the values of a RSP by issuing the command `sccp rsp modify` with appropriate parameters as described below:

```
Name
    sccp rsp modify

SYNOPSIS
    sccp rsp modify <id> <remote-spc> <rspc-flag> <mask> stackname <stack-name>

DESCRIPTION
    This command is used to modify the values of a Remote Signaling Point previously
    defined.

PARAMETERS

    Standard Parameters

    <id>          -   Identifier of the Remote Signaling Point to be
                      modified.

    <remote-spc>      -   The Remote Signaling Point

    <rspc-flag>       -   32 bit value. Not used for now.
                      Reserved for future

    <mask>            -   32 bit value. Not used for now.
                      Reserved for future

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

7.4.23. Delete a Remote Signaling Pointcode

Using CLI

You can delete a RSP by issuing the command `sccp rsp delete` with appropriate parameters as described below:

```
Name
    sccp rsp delete

SYNOPSIS
    sccp rsp delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to delete a Remote Signaling Point.

PARAMETERS

    Standard Parameters

    <id>      -   The identifier of the Remote Signaling Point that is being
                  deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp rsp delete 1

    The above command will delete the Remote Signaling Point identified by the
    number '1'.
```

Using GUI

Procedure: Delete a RSP using GUI

1. Navigate to the 'RSP' tab in the SCCP Management Unit window and locate the row corresponding to the RSP you wish to delete.
2. In the row corresponding to the identified RSP, click on the red 'x' button in the actions column to delete that RSP.

7.4.24. View all configured Remote Sub-Systems (RSS)

Using CLI

You can view the details of all configured Remote Sub-Systems by issuing the command `sccp rss show` as described below:

```
Name
    sccp rss show

SYNOPSIS
    sccp rss show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Remote Sub-Systems.
    If an <id> is specified in the command, it will only display the details of the
    Remote Sub-System identified by the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>      -   The id of the Remote Sub-System whose details are to be
                  displayed. If this parameter is not specified, the details
                  of all configured Remote Sub-Systems will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'RSS' tab. Here you can view a list of all the configured Remote Sub-Systems. Every correctly configured RSS will be displayed in a row along with the defined values.

7.4.25. Create a new Remote Sub-System

Using CLI

You can create a new RSS by issuing the command `sccp rss create` with appropriate parameters as described below:

```
Name
    sccp rss create

SYNOPSIS
    sccp rss create <id> <remote-spc> <remote-ssn> <rss-flag>
    <mark-prohibited-when-spc-resuming> stackname <stack-name>

DESCRIPTION
    This command is used to define a new Remote Sub-System. Each Remote Sub-System
    that SCCP can communicate with must be configured using this command. You must
    ensure that the Remote Signaling Point is configured prior to issuing this
    command.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this Remote
                    Sub-System.

    <remote-spc>      -   The Remote Signaling Point where this Remote
                    Sub-System is being deployed.

    <remote-ssn>      -   The Remote Sub-System number.

    <rss-flag>        -   32 bit value. Not used for now.
                    Reserved for future.

    Optional Parameters

    <mark-prohibited-when-spc-resuming>
                -   Possible value: prohibitedWhenSpcResuming.
                    When this parameter is specified, the configured
                    subsystem is marked as prohibited when its
                    corresponding signaling point code has been
                    resumed.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp rss create 1 6477 8 0 prohibitedWhenSpcResuming
```

Using GUI

Procedure: Create a new RSS using GUI

1. Navigate to the 'RSS' tab in the SCCP Management window and click on the 'Create RSS' button. This will launch a new pop-up 'Create Remote Sub-System'.

2. In the 'Create Remote Sub-System' pop-up, add details of the new RSS being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new RSS will be created with parameters as specified. If there is an error in creating the RSS then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Remote Sub-System' pop-up.

7.4.26. Modify a Remote Signaling Sub-System

Using CLI

You can modify the values of a RSS by issuing the command `sccp rss modify` with appropriate parameters as described below:

```
Name
    sccp rss modify

SYNOPSIS
    sccp rss modify <id> <remote-spc> <remote-ssn> <rss-flag>
    <mark-prohibited-when-spc-resuming> stackname <stack-name>

DESCRIPTION
    This command is used to modify the values of a Remote Sub-System previously
    defined.

PARAMETERS

    Standard Parameters

    <id>          -   Identifier of the Remote Sub-System to be
                    modified.

    <remote-spc>      -   The Remote Signaling Point where this Remote
                    Sub-System is deployed.

    <remote-ssn>      -   The Remote Sub-System number.

    <rss-flag>        -   32 bit value. Not used for now.
                    Reserved for future.

    Optional Parameters

    <mark-prohibited-when-spc-resuming>
                -   Possible value: prohibitedWhenSpcResuming.
                    When this parameter is specified, the configured
                    subsystem is marked as prohibited when its
                    corresponding signaling point code has been
                    resumed.

    Optional Parameters

    <mark-prohibited-when-spc-resuming>
                -   Possible value: prohibitedWhenSpcResuming.
                    When this parameter is specified, the configured
                    subsystem is marked as prohibited when its
                    corresponding signaling point code has been
                    resumed.

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

7.4.27. Delete a Remote Signaling Sub-System

Using CLI

You can delete a RSS by issuing the command `sccp rss delete` with appropriate parameters as described below:

```
Name
    sccp rss delete

SYNOPSIS
    sccp rss delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to delete a Remote Sub-System.

PARAMETERS

    Standard Parameters

    <id>      -   The identifier of the Remote Sub-System that is being
                  deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    sccp rss delete 1

    The above command will delete the Remote Sub-System identified by the number '1'.
```

Using GUI

Procedure: Delete a RSS using GUI

1. Navigate to the 'RSS' tab in the SCCP Management Unit window and locate the row corresponding to the RSS you wish to delete.
2. In the row corresponding to the identified RSS, click on the red 'x' button in the actions column to delete that RSS.

7.4.28. View all configured Long Message Rules (LMR)

Using CLI

You can view the details of all configured Long Message Rules by issuing the command `sccp lmr show` as described below:

```
Name
    sccp lmr show

SYNOPSIS
    sccp lmr show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Long Message Rules.
    If an <id> is specified in the command, it will only display the details of the
    Long Message Rule identified by the value of the 'id' specified.

PARAMETERS

    Optional Parameters

    <id>          -   The id of the Long Message Rule whose details are to be
                      displayed. If this parameter is not specified, the details
                      of all configured Long Message Rules will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'LMR' tab. Here you can view a list of all the configured Long Message Rules. Every correctly configured LMR will be displayed in a row along with the defined values.

7.4.29. Create a new Long Message Rule

Using CLI

You can create a new LMR by issuing the command `sccp lmr create` with appropriate parameters as described below:

```
Name
    sccp lmr create

SYNOPSIS
    sccp lmr create <id> <first-spc> <last-spc> <long-message-rule-type>
    stackname <stack-name>

DESCRIPTION
    This command is used to define a new Long Message Rule. Long message rules specify
    which message types (UDT/XUDT/LUDT) will be used for outgoing message encoding
    depends on dpc. If long message rules are not configured only UDT messages
    will be used.

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this
                    Long Message Rule.

    <first-spc>       -   The first value of the remote signaling
                    point code range, for which this Long message Rule
                    will apply.

    <last-spc>        -   The last value of the remote signaling point code
                    range. If Long message rule specifies a single
                    signaling point code, this value must be equal to
                    first-spc.

    <long-message-rule-type>- Message types used for the remote signaling
                    point codes range.
                    Possible values : udt, xudt, ludt and ludt_segm.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    sccp lmr create 1 201 201 xudt

    sccp lmr create 2 230 239 udt
```

Using GUI

Procedure: Create a new LMR using GUI

1. Navigate to the 'LMR' tab in the SCCP Management window and click on the 'Create LMR' button. This will launch a new pop-up 'Create Long Message Rule'.

2. In the 'Create Long Message Rule' pop-up, add details of the new LMR being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new LMR will be created with parameters as specified. If there is an error in creating the LMR then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Long Message Rule' pop-up.

7.4.30. Modify a Long Message Rule

Using CLI

You can modify the values of a LMR by issuing the command `sccp lmr modify` with appropriate parameters as described below:

```
Name
    sccp lmr modify

SYNOPSIS
    sccp lmr modify <id> <first-spc> <last-spc> <long-message-rule-type>
    stackname <stack-name>

DESCRIPTION
    This command is used to modify the values of a Long Message Rule previously
    defined.

PARAMETERS

    Standard Parameters

    <id>          -   Identifier of the Long Message Rule to be
                    modified.

    <first-spc>       -   The first value of the remote signaling
                    point code range, for which this Long message Rule
                    will apply.

    <last-spc>        -   The last value of the remote signaling point code
                    range. If Long message rule specifies a single
                    signaling point code, this value must be equal to
                    first-spc.

    <long-message-rule-type>- Message types used for the remote signaling point
                    codes range.
                    Possible values : udt, xudt, ludt and ludt_segm.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

7.4.31. Delete a Long Message Rule

Using CLI

You can delete a LMR by issuing the command `sccp lmr delete` with appropriate parameters as described below:

```
Name
    sccp lmr delete

SYNOPSIS
    sccp lmr delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to delete a Long Message Rule.

PARAMETERS

    Standard Parameters

    <id>      -   The identifier of the Long Message Rule that is being
                  deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    sccp lmr delete 1

    The above command will delete the Long Message Rule identified by the number '1'.
```

Using GUI

Procedure: Delete a Long Message Rule using GUI

1. Navigate to the 'LMR' tab in the SCCP Management Unit window and locate the row corresponding to the LMR you wish to delete.
2. In the row corresponding to the identified LMR, click on the red 'x' button in the actions column to delete that LMR.

7.4.32. View all configured Concerned Signaling Point Codes (CSP)

Using CLI

You can view the details of all configured Concerned Signaling Point Codes by issuing the command `sccp csp show` as described below:

```
Name
    sccp csp show

SYNOPSIS
    sccp csp show id <id> stackname <stack-name>

DESCRIPTION
    This command is used to view the details of all configured Concerned Signaling
    Point Codes. If an <id> is specified in the command, it will only display the
    details of the Concerned Signaling Point Code identified by the value of the 'id'
    specified.

PARAMETERS

    Optional Parameters

    <id>          -   The id of the Concerned Signaling Point Code whose details
                      are to be displayed. If this parameter is not specified,
                      the details of all configured Concerned Signaling Point
                      Codes will be displayed.

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.
```

Using GUI

Navigate to the specific SCCP Management unit and switch to the 'CSP' tab. Here you can view a list of all the configured Concerned Signaling Point Code. Every correctly configured CSP will be displayed in a row along with the defined values.

7.4.33. Create a new Concerned Signaling Point Code

Using CLI

You can create a new CSP by issuing the command `sccp csp create` with appropriate parameters as described below:

```
Name
    sccp csp create

SYNOPSIS
    sccp csp create <id> <spc> stackname <stack-name>

DESCRIPTION
    This command is used to define a new Concerned Signaling Point Code. Concerned
    signaling point codes define a DPC list that will be notified when local SSN is
    registered (SSA messages) or unregistered (SSP messages).

PARAMETERS

    Standard Parameters

    <id>          -   A unique number to identify this
                      Concerned Signaling Point Code.

    <spc>         -   The Remote Signaling Point Code,
                      which will be notified.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.


EXAMPLES
    sccp csp create 1 201

    sccp csp create 2 202
```

Using GUI

Procedure: Create a new CSP using GUI

1. Navigate to the 'CSP' tab in the SCCP Management window and click on the 'Create CSP' button. This will launch a new pop-up 'Create Concerned Signaling Point Code'.
2. In the 'Create Concerned Signaling Point Code' pop-up, add details of the new CSP being defined. You must ensure that you fill in all the mandatory parameters. For definition of these parameters, please refer to the description of the CLI command for the same in the preceding section.
3. Verify the details entered and then click on the 'Create' button. A new CSP will be created with parameters as specified. If there is an error in creating the CSP then you will find the details of the error in the Management Console Log section below.
4. Click on the 'Close' button to close the 'Create Concerned Signaling Point Code' pop-up.

7.4.34. Modify a Concerned Signaling Point Code

Using CLI

You can modify the values of a CSP by issuing the command `sccp csp modify` with appropriate parameters as described below:

```
Name
    sccp csp modify

SYNOPSIS
    sccp csp modify <id> <spc> stackname <stack-name>

DESCRIPTION
    This command is used to modify the values of a Concerned Signaling Point Code
    previously defined.

PARAMETERS

    Standard Parameters

    <id>          -   Identifier of the Concerned Signaling Point Code
                    to be modified.

    <spc>         -   The Remote Signaling Point Code, which will be
                    notified.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.
```

7.4.35. Delete a Concerned Signaling Point Code

Using CLI

You can delete a CSP by issuing the command `sccp csp delete` with appropriate parameters as described below:

```
Name
    sccp csp delete

SYNOPSIS
    sccp csp delete <id> stackname <stack-name>

DESCRIPTION
    This command is used to delete a Concerned Signaling Point Code.

PARAMETERS

    Standard Parameters

    <id>      -   The identifier of the Concerned Signaling Point Code
                  that is being deleted.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                      If not passed, the first stack configured in ShellExecutor
                      will be used.

EXAMPLES
    sccp csp delete 1

    The above command will delete the Concerned Signaling Point Code identified by the
    number '1'.
```

Using GUI

Procedure: Delete a Concerned Signaling Point Code using GUI

1. Navigate to the 'CSP' tab in the SCCP Management Unit window and locate the row corresponding to the CSP you wish to delete.
2. In the row corresponding to the identified CSP, click on the red 'x' button in the actions column to delete that CSP.

7.5. TCAP Management

Transaction Capabilities Application Part , from ITU-T recommendations Q.771-Q.775 or ANSI T1.114 is a protocol for Signalling System 7 networks. Its primary purpose 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.

7.5.1. Using CLI

You can manage all TCAP stack properties through the Command Line Interface by using the `tcap`

command.

7.5.2. Using GUI

The GUI will allow you to manage your TCAP configurations efficiently using a user-friendly interface. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'TCAP' link in the left panel. The main panel will display the names of all configured TCAP Management units. To configure or view the settings of a particular TCAP Management Unit you must click on the name of that unit. The GUI will look similar to the figure below.

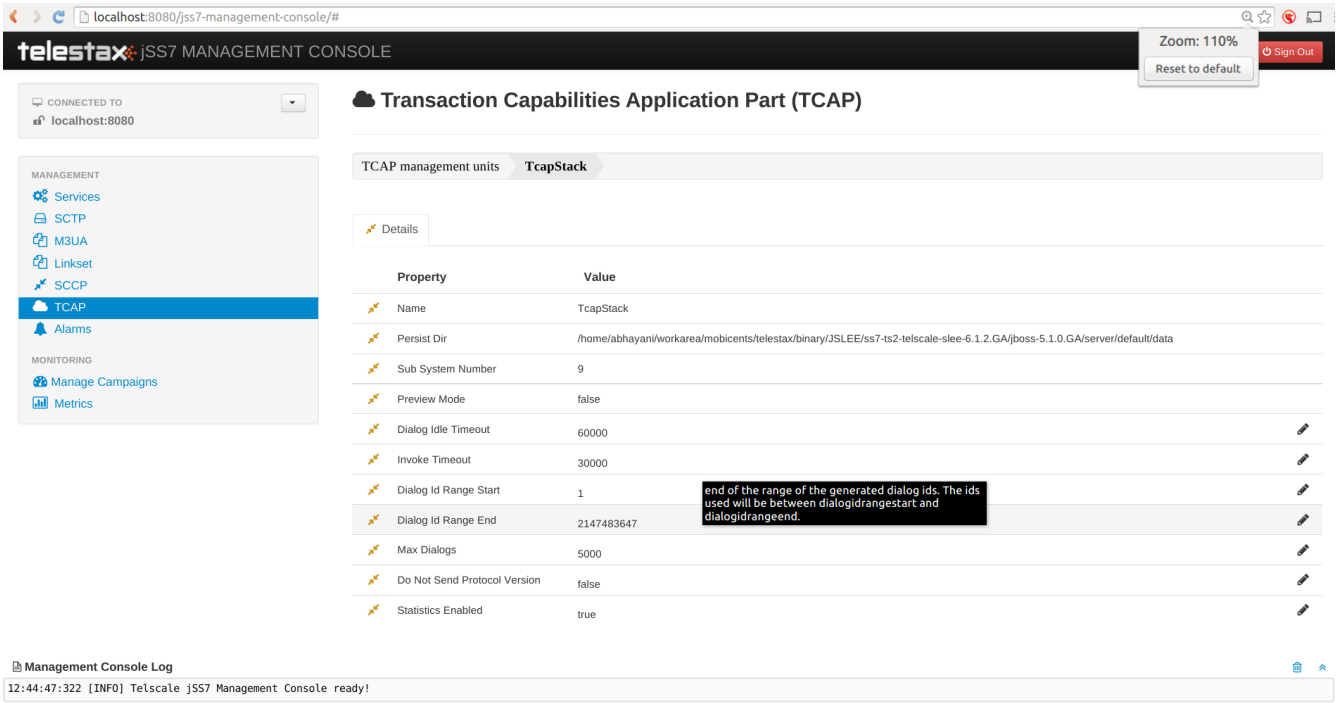


Figure 26. GUI - TCAP Management

The first tab will display the properties of the TCAP Management unit. These details displayed here are fetched from the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name. These properties can be modified here in the GUI. To modify them you must click the pencil, change value and save. The GUI will then display the modified values.

7.5.3. TCAP stack properties

Dialog Idle Timeout

Using CLI

You can set the 'dialogidletimeout' by issuing the command `tcap set dialogidletimeout` with appropriate parameters as described below. You can verify this by issuing the command `tcap get dialogidletimeout` which will display the value set for this property.

```
Name
    tcap set dialogidletimeout

SYNOPSIS
    tcap set dialogidletimeout <dialogidletimeout> stackname <stack-name>

DESCRIPTION
    Sets millisecond value for dialog timeout. It specifies how long
    dialog can be idle - not receive/send any messages.
    When a timeout occurs the method `TCListener.onDialogTimeout()` will be
    invoked. If a TCAP-User does not invoke `Dialog.keepAlive()` inside
    the method `TCListener.onDialogTimeout()`, the TCAP Dialog will be
    released.

PARAMETERS

    Standard Parameters

    <dialogidletimeout>   -   Timeout in milliseconds.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set dialogidletimeout 30000
```

Using GUI

On TCAP management page, click on pencil against the 'Dialog Idle Timeout' property and text box becomes editable. Change value and save.

Dialog Id Range End

TCAP stack can be configured to use a range of local DialogId values. You may install a set of TCAP Stack instances with different DialogId ranges. These ranges can be used for loadsharing of SS7 traffic between the TCAP instances. All the outgoing Dialogs will have id starting with `dialogIdRangeStart`. This value of `dialogIdRangeStart` cannt be greater than `dialogIdRangeEnd`. In addition, the value of `dialogIdRangeEnd - dialogIdRangeStart` must always be less than the value of `maxDialogs`.

Using CLI

You can set the 'dialogidrangeend' by issuing the command `tcap set dialogidrangeend` with appropriate parameters as described below. You can verify this by issuing the command `tcap get dialogidrangeend` which will display the value set for this property.

```
Name
    tcap set dialogidrangeend

SYNOPSIS
    tcap set dialogidrangeend <dialogidrangeend> stackname <stack-name>

DESCRIPTION
    End of the range of the generated dialog ids. The id's used will be
    between dialogidrangestart and dialogidrangeend.

PARAMETERS

    Standard Parameters

    <dialogidrangeend>    -   Dialog id range end.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set dialogidrangeend 30000000
```

Using GUI

On TCAP management page, click on pencil against the 'Dialog Id Range End' property and text box becomes editable. Change value and save.

Dialog Id Range Start

Using CLI

You can set the 'dialogidrangestart' by issuing the command `tcap set dialogidrangestart` with appropriate parameters as described below. You can verify this by issuing the command `tcap get dialogidrangestart` which will display the value set for this property.

```
Name
    tcap set dialogidrangestart

SYNOPSIS
    tcap set dialogidrangestart <dialogidrangestart> stackname <stack-name>

DESCRIPTION
    Start of the range of the generated dialog ids. The id's used will be
    between dialogidrangestart and dialogidrangeend.

PARAMETERS

    Standard Parameters

    <dialogidrangestart>  -   Dialog id range start.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set dialogidrangestart 1
```

Using GUI

On TCAP management page, click on pencil against the 'Dialog Id Range Start' property and text box becomes editable. Change value and save.

Do Not Send Protocol Version

Using CLI

You can set the 'donotsendprotocolversion' by issuing the command `tcap set donotsendprotocolversion` with appropriate parameters as described below. You can verify this by issuing the command `tcap get donotsendprotocolversion` which will display the value set for this property.

```
Name
    tcap set donotsendprotocolversion

SYNOPSIS
    tcap set donotsendprotocolversion <true | false> stackname <stack-name>

DESCRIPTION
    If set to true Protocol Version is not send in User Data part of Dialog

PARAMETERS

    Standard Parameters

    <donotsendprotocolversion>    -   If true doesn't send the protocol version

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set donotsendprotocolversion false
```

Using GUI

On TCAP management page, click on pencil against the 'Do Not Send Protocol Version' property and text box becomes editable. Change value and save.

Invoke Timeout

Using CLI

You can set the 'invoketimeout' by issuing the command `tcap set invoketimeout` with appropriate parameters as described below. You can verify this by issuing the command `tcap get invoketimeout` which will display the value set for this property.

```
Name
    tcap set invoketimeout

SYNOPSIS
    tcap set invoketimeout <invoketimeout> stackname <stack-name>

DESCRIPTION
    Sets the Invoke timeout for this invoke. This property specifies,
    by default, how long Invoke will wait for a response from a peer
    before a timeout occurs.
    If a TCAP-User does not specify a custom Invoke timeout when
    sending a new Invoke, this default value will be used for outgoing
    Invoke timeout. When this timeout occurs `TCListener.onInvokeTimeout()`
    will be invoked.
    invoketimeout should always be less than dialogidletimeout.
    This parameter affects if we use TCAP stack as the upperst level or
    we have implemented our own stack that reuses TCAP stack.
    Restcomm MAP and CAP stacks overrides this parameter at their levels
    and this parameter deos not affect these stacks.

PARAMETERS

    Standard Parameters

    <invoketimeout>       -   Sets the Invoke timeout in milliseconds

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set invoketimeout 30000
```

Using GUI

On TCAP management page, click on pencil against the 'Invoke Timeout' property and text box becomes editable. Change value and save.

Max Dialogs

Using CLI

You can set the 'maxdialogs' by issuing the command `tcap set maxdialogs` with appropriate parameters as described below. You can verify this by issuing the command `tcap get maxdialogs` which will display the value set for this property.

```
Name
    tcap set maxdialogs

SYNOPSIS
    tcap set maxdialogs <maxdialogs> stackname <stack-name>

DESCRIPTION
    Sets the maximum number of dialogs allowed to be alive at a given
    time. If not set, a default value of 5000 dialogs will be used.
    If stack ranges provided, maximum number dialogs naturally cannot
    be greater than the provided range, thus, it will be normalized to
    range delta (end - start).

PARAMETERS

    Standard Parameters

    <maxdialogs>  -   Sets the maximum concurrent dialogs alive at any given
                        point in time.

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set maxdialogs 30000000
```

Using GUI

On TCAP management page, click on pencil against the 'Max Dialogs' property and text box becomes editable. Change value and save.

Preview Mode

You can modify the settings for the parameter 'previewmode' only when the TCAP Stack is not running. In addition, this parameter cannot be modified through the CLI or GUI. You will have to invoke the setter function directly from the source code.

If you are using the JBoss Application Server, then you can set this parameter by adding a property (as shown below) to the XML descriptor file *jboss-beans.xml*, which is located at *$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF*, where `profile_name` is the server profile name.

```
/*Add property for the parameter 'previewmode' to jboss-beans.xml file and specify
true or false*/
<property name="previewMode">true</property>
```

The current settings of the parameter can be viewed in the GUI or by invoking the appropriate CLI command as described below.

Using CLI

You can retrieve the current settings of the parameter 'previewmode' by issuing the command `sctp get previewmode`. However as explained above, you cannot modify the settings through the CLI.

```
Name
    tcap get previewmode

SYNOPSIS
    tcap get previewmode

DESCRIPTION
    This command is used to retrieve the current settings of the parameter
    'previewMode'. The 'previewMode' parameter is used for special processing
    mode.

    When Preview Mode is set to true:
    - In TCAP level the stack only listens for incoming messages and
    sends nothing.
    - Methods like send(), close(), sendComponent() and other such methods
    do nothing.
    - A TCAP Dialog is temporary. The TCAP Dialog is discarded after
    any incoming message like TC-BEGIN or TC-CONTINUE has been processed.
    - For any incoming messages (including TC-CONTINUE, TC-END, TC-ABORT) a new
    TCAP Dialog is created (and then deleted).
    - There are no timers and timeouts.

    The settings of this parameter can be modified only when the TCAP Stack is
    not running. To modify this parameter you must invoke the setter function
    directly from the code or if you are using the JBoss AS, you can add a
    property to the XML descriptor file jboss-beans.xml. You cannot change the
    settings through the CLI.
```

Using GUI

In the TCAP management page, you can view the current settings of the 'Preview Mode' property. But as explained above, you cannot change the settings in the GUI. For more details about this parameter, refer to the detailed description about the parameter in the above section for CLI.

Statistics Enabled

Using CLI

You can set the 'statisticsenabled' by issuing the command `tcap set statisticsenabled` with appropriate parameters as described below. You can verify this by issuing the command `tcap get statisticsenabled` which will display the value set for this property.

```
Name
    tcap set statisticsenabled

SYNOPSIS
    tcap set statisticsenabled <true | false> stackname <stack-name>

DESCRIPTION
    If set to true, statistics is enabled. Its recommended to keep this off
    for better performance and enabled statistics only when needed.

PARAMETERS

    Standard Parameters

    <statisticsenabled>   -   If true, statistics is enabled

    Optional Parameters

    <stack-name>  -   Name of the stack on which this command is executed.
                    If not passed, the first stack configured in ShellExecutor
                    will be used.

EXAMPLES
    tcap set statisticsenabled false
```

Using GUI

On TCAP management page, click on pencil against the 'Statistics Enabled' property and text box becomes editable. Change value and save.

7.6. Statistics

The GUI will allow you to create campaigns of fixed duration for gathering statistics data. Campaign allows to select time period over which these statistics have been gathered (in hours, minutes and seconds). Once Campaign is defined, the statistics can be observed by clicking newly created campaign name or you can also navigate to Metrics (click Metrics on left panel) to get graph of statistics.

7.6.1. Create Campaign

To create new campaign open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'Manage Campaigns' link in the left panel. The main panel will display the names of all existing campaigns and also button to create new campaign. The GUI will look similar to the figure below.

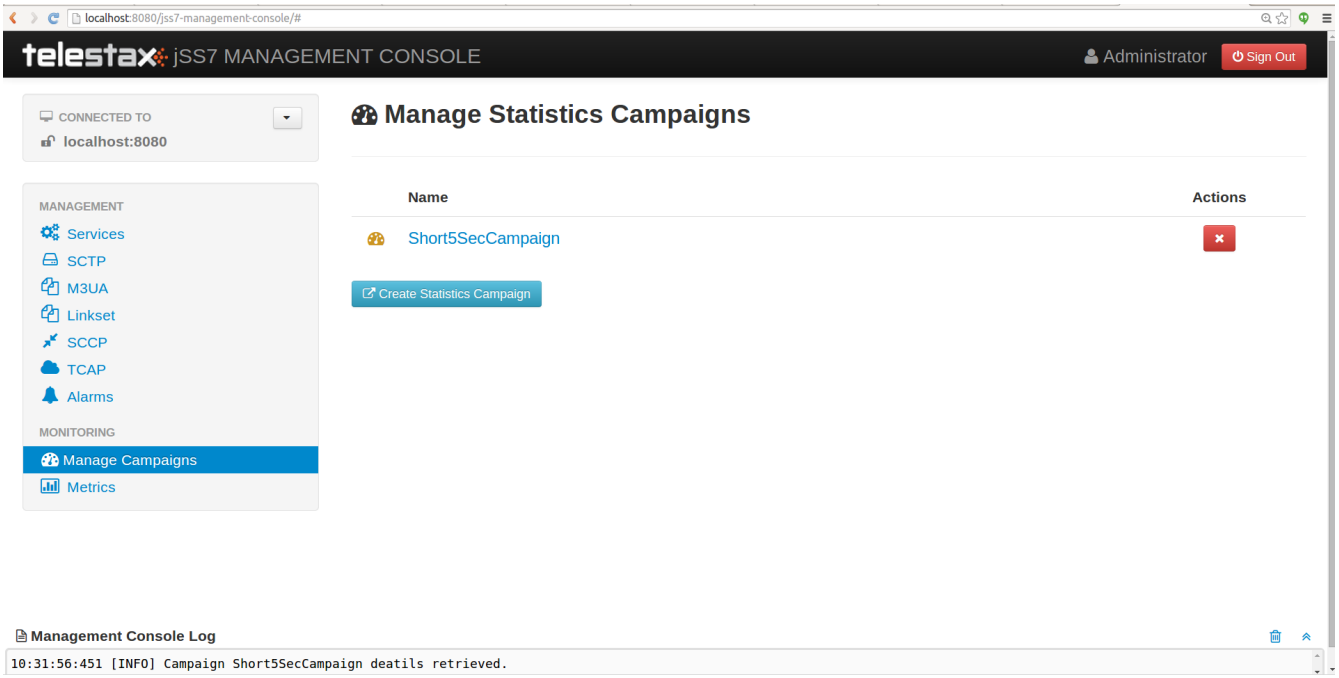


Figure 27. GUI - Campaigns

Click on 'Create Statistics Campaign' button to create new campaign. Select the stack from 'Counter Definition Set Name' drop down on which you want to define new campaign. Next select the time period from 'Duration' drop down and enter unique 'Campaign Name'.

The stack on which new campaign is defined must have set 'Statistics Enabled' property to true

7.6.2. View Campaigns

The GUI will allow you to view existing campaigns. On the main panel click campaign name. The GUI will look similar to the figure below and is divided into tabs.

The first tab will display the properties of the campaign. Second tab explains all the counters in this campaigns and their definition. Last tab provides the values for each of these counters. Last tab also displays the 'Start Time' and 'End Time' representing time duration for which sample was collected.

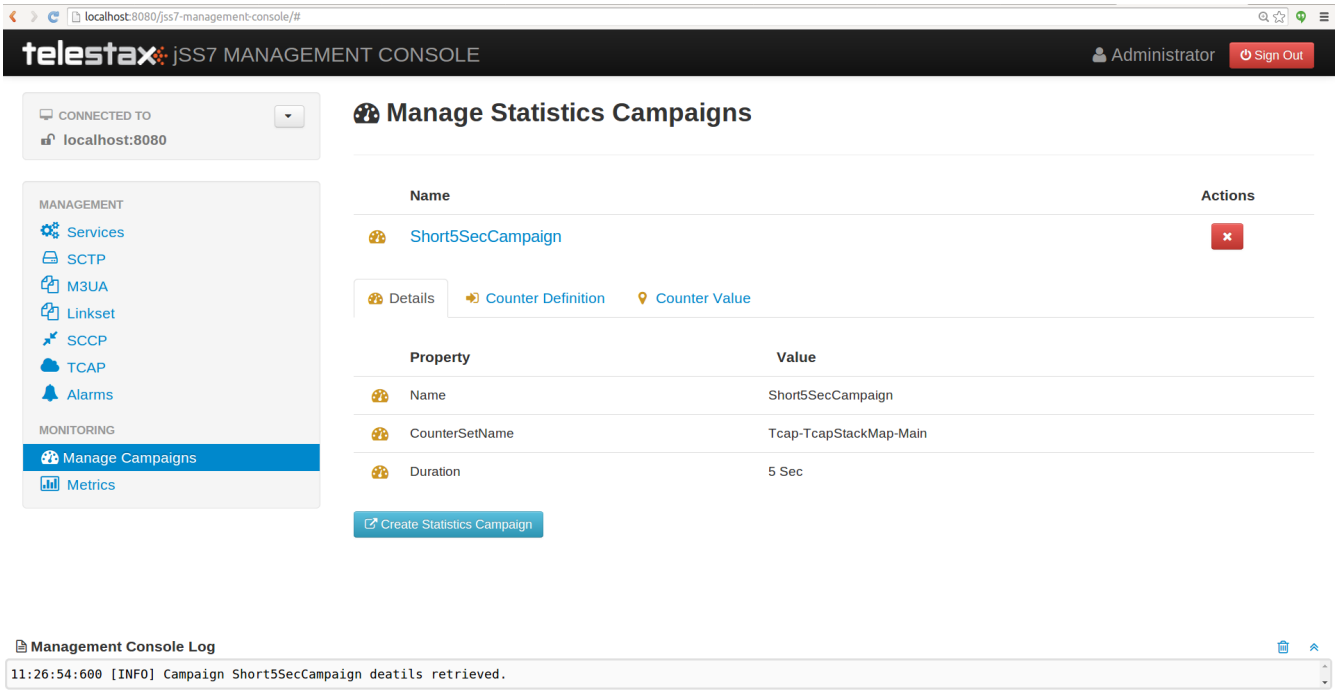


Figure 28. GUI - Campaigns View

Restcomm jSS7 doesn't persist the statistics, hence the data collected for campaign period refresh's every defined 'Duration'. User must refresh the page every 'Duration' period to gather statistics data for previous time period.

Nevertheless you can also click on 'Metrics' link on left panel, select the Campaign and observe the statistics graph. The metrics page gathers data from time page was loaded till user navigates away. Hence graph will show historic data from point page was loaded.

7.6.3. Logging Stats

The GUI will allow you to view stats in real time. But the stats are not stored in the Database for you to analyse at a later point of time. However Restcomm jSS7 gives you an option to have the stats logged every refresh period for all the existing campaigns. You can look at the log files at any point of time for analysing or understanding the performance.

If you wish to have stats logged, you must configure the settings in the file *jboss-5.1.0.GA/server/default/conf/jboss-log4j.xml*. If you are running the platform as standalone, then you should configure in *log4j.xml*.

Logging stats to main server log file

If you wish to have the stats logged in the main server log file located at *jboss-5.1.0.GA/server/default/log/server.log*, then you must add a new category to the *jboss-log4j.xml* as shown below and set priority value to "DEBUG".

```
<category name="org.mobicents.protocols.ss7.oam.common.statistics.StatsPrinter"
additivity="false">
        <priority value="DEBUG" />
</category>
```

Logging stats to a separate stats log file

If you wish to have the stats logged to a separate log file located at *jboss-5.1.0.GA/server/default/log/stats.log*, then you must add a new appender to the *jboss-log4j.xml* and change category as shown below. You must set the priority value to "DEBUG".

```
<appender name="STATS" class="org.jboss.logging.appender.DailyRollingFileAppender">
<errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
<param name="File" value="$
{jboss.server.home.dir}
/log/stats.log"/>
<param name="Append" value="true"/>
<param name="MaxFileSize" value="500KB" />
<param name="MaxBackupIndex" value="1" />
<param name="Threshold" value="DEBUG"/>
<param name="DatePattern" value="'.'yyyy-MM-dd"/>
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
</layout>
</appender>
<category name="org.mobicents.protocols.ss7.oam.common.statistics.StatsPrinter"
additivity="false">
<priority value="DEBUG" />
<appender-ref ref="STATS"/>
</category>
```

As of today Restcomm jSS7 exposes statistics only for TCAP. Future releases will allow statistics for M3UA, SCCP, MAP and CAP also.

7.7. Alarms

The GUI will allow you to view the Alarms (Critical, Major, Minor, Warning) in the system and filter them based on the severity. Open a Web Browser and navigate to http://localhost:8080/jss7-management-console/. Click on the 'Alarms' link in the left panel. The main panel will display all the currently raised alarms. You can filter the view by using the checkboxes at the top to select/deselect the severity options and accordingly filter the display.

You will notice that as soon as the problem is resolved, the corresponding alarm will disappear from the view. In future releases, there will be an option to persist this alarms data and hence you will be able to view a history of alarms raised and resolved. But as of now, only currently active alarms will be displayed in this window.

Chapter 8. M3UA

8.1. Restcomm Signaling Gateway M3UA Stack

M3UA is a client-server protocol supporting the transport of any SS7 MTP3-User signaling (e.g. ISUP and SCCP messages) over IP. M3UA is defined by the IETF SIGTRAN working group in RFC 4666. Restcomm M3UA Stack can be used on the Application Server side or on the Signaling Gateway side or can also be used in peer-to-peer mode IPSP.

M3UA uses the Restcomm SCTP (Stream Control Transmission Protocol) Stack.

Note

Restcomm SCTP Stack uses Java SCTP layer which is only available from JDK 7 onwards.

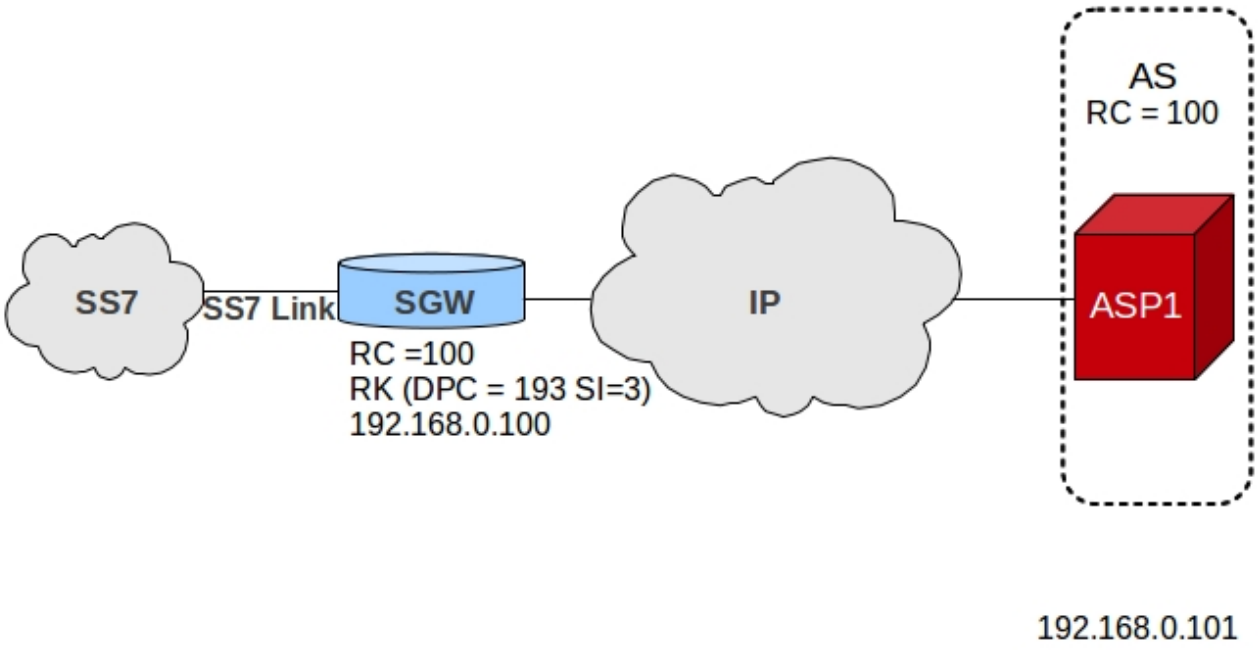
8.1.1. M3UA Load Balancing

Load Balancing can be configured by using the M3UA configuration parameter useLsbForLinksetSelection defined in the *jboss-beans.xml* file. Refer to Configuring M3UA for more details on configuring this.

Depending on the configuration (useLsbForLinksetSelection set to true or false), either the least significant bit or the most significant bit of SLS is used for load balancing between two Application Servers. The remaining bits in the SLS are used for load balancing between ASPs within each AS.

8.1.2. Restcomm M3UA Stack on the Application Server side

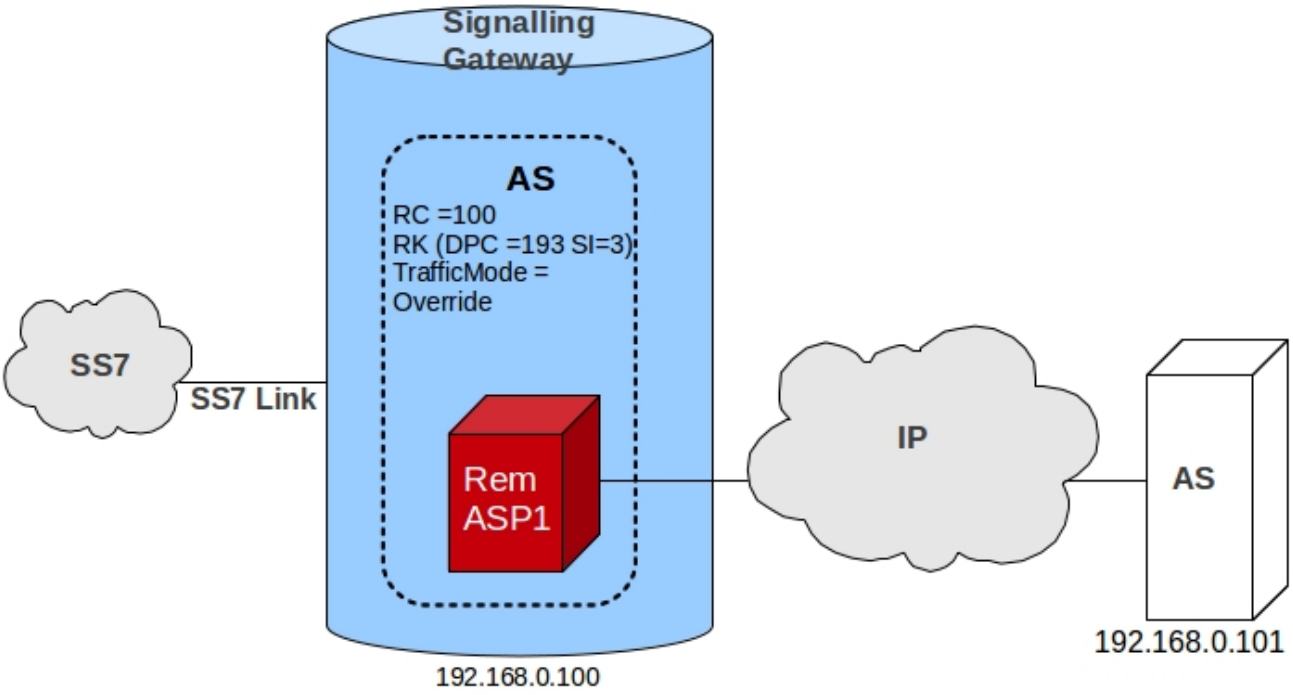
The figure below demonstrates the basic functionality of the Restcomm M3UA Stack when configured as an Application Server (AS) that will communicate with an External Signaling Gateway.



To use M3UA Stack as an AS, the Routing Context (RC) may not be known and is optional. Refer to Configuring M3UA for help in configuring M3UA Stack as an AS.

8.1.3. Restcomm M3UA Stack on the Signaling Gateway side

The figure below demonstrates the basic functionality of the Restcomm M3UA Stack when configured as a Signaling Gateway (SG). The Restcomm Signaling Gateway provides the Nodal Interworking Function (NIF) that allows SS7 Signaling (SCCP/ISUP) to be inter-worked into the M3UA/IP Network.



Restcomm M3UA Stack used on the SG side will share a common point code with a set of M3UA Application Servers. You can configure the M3UA stack on SG side in one of the two traffic modes: Loadbalance or Override. Broadcast traffic mode is not supported. Refer to Configuring Restcomm Signaling Gateway for instructions on configuring M3UA Stack as SG. Restcomm M3UA Stack used on SG side doesn't support routing key management messages. The Routing Key should be provisioned statically using the management console.

8.1.4. Restcomm M3UA Stack as IPSP

An IPSP is essentially the same as an ASP, except that it uses M3UA in a point-to-point fashion. Conceptually, an IPSP does not use the services of a Signalling Gateway node.

8.2. Route

Before you can transfer Payload Data from M3UA-User to a peer, you must define the route based on Destination Point Code (DPC), Originating Point Code (OPC) and Service Indicator (SI). For details on how to add a new Route, please refer to Create a new M3UA Route.

While DPC is mandatory and should be an actual value, OPC and SI can be -1 indicating wild card. The table below shows an example of a routeset table. The routeset table contains routesets for all the possible destinations that can be reached. This table is searched to find a match for the DPC:OPC:SI to be routed. If a match is found in the list of Application Servers, an AS is chosen from the available routes associated with the routeset. If an AS is not found, SI is substitued with -1 (DPC:OPC:-1) and a match is searched for again. If a match still is not found, OPC is substituted with -1 (DPC:-1:-1) and the table is searched again for a match. If there is still no matching AS, MSU is dropped and routing does not take place.

Table 4. Routesets

DPC	OPC	SI	AS Name
2	3	2	AS1, AS2
2	3	-1	AS1, AS2
4	-1	-1	AS3

8.3. Load Sharing

M3UA load-balancing makes use of the SLS field of the Protocol Data carried in the Payload Data Message. M3UA can be configured as below:

- load-balance between Application Servers (AS) for a given route. For any given route, there can be a maximum of 16 ASs defined.
- load-balance within an AS and between Application Server Processes (ASP). A maximum 128 ASPs can be defined within an AS for load-sharing.

For even distribution of messages in the network it is recommended that you define an even number of ASs for a route for load sharing. You must ensure that you take proper care in deciding on a number of ASs for a route and number of ASPs within an AS depending on the routing label format of MSU.

M3UA can be configured to use either the highest or lowest bits of the SLS for AS selection. The number of SLS bits used for AS selection depends on the number of maximum AS defined for a route. The remaining bits are used for ASP selection.

Table 5. Routesets

Max AS for Route	Number of SLS bits used for AS	Max ASPs that can be used within an AS
1 or 2	1	128
3 or 4	2	64
5 to 8	3	32
9 to 16	4	16

8.4. M3UA Internal State Machine

Restcomm M3UA Stack maintains finite state machine (FSM) for each ASP and AS. Its important to understand these state's to troubleshoot the M3UA handshake messages exchanged

The FSM for ASP at AS side (or IPSPS client side) is referred to as LocalFSM and FSM for AS at AS side (or IPSP client side) is referred to as PeerFSM. Similarly FSM for ASP at SGW side (or IPSP server side) is referred to as PeerFSM and FSM for AS at SGW side (or IPSP server side) is referred to as LocalFsm.

Figure below shows the various state of ASP at AS and SGW side. In below example only Single Exchange of messages are considered for handshake mechanism

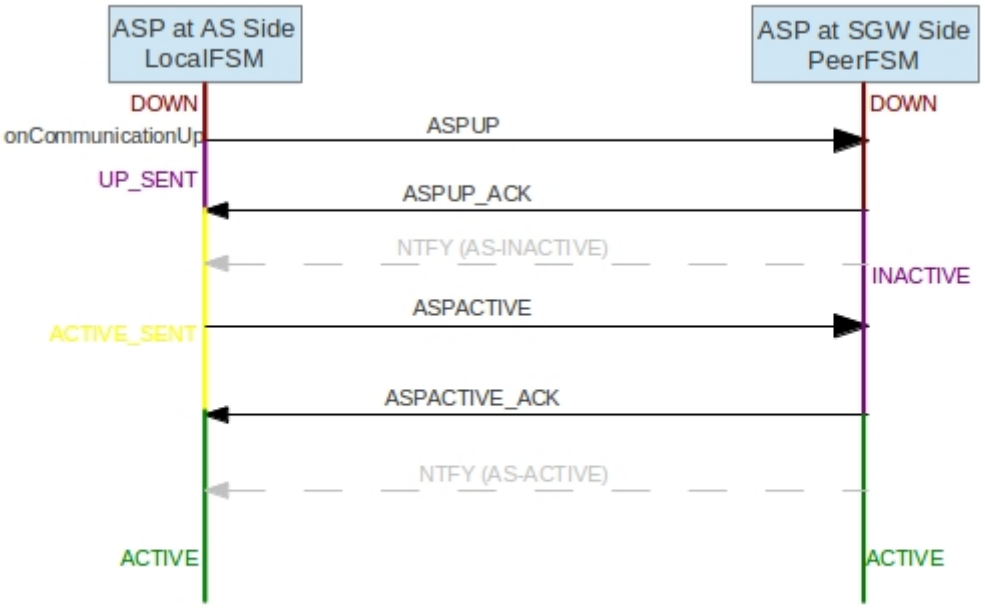


Figure 29. Restcomm M3UA Stack ASP state machine

Figure below shows the various state of AS at AS and SGW side. In below example only Single Exchange of messages are considered for handshake mechanism

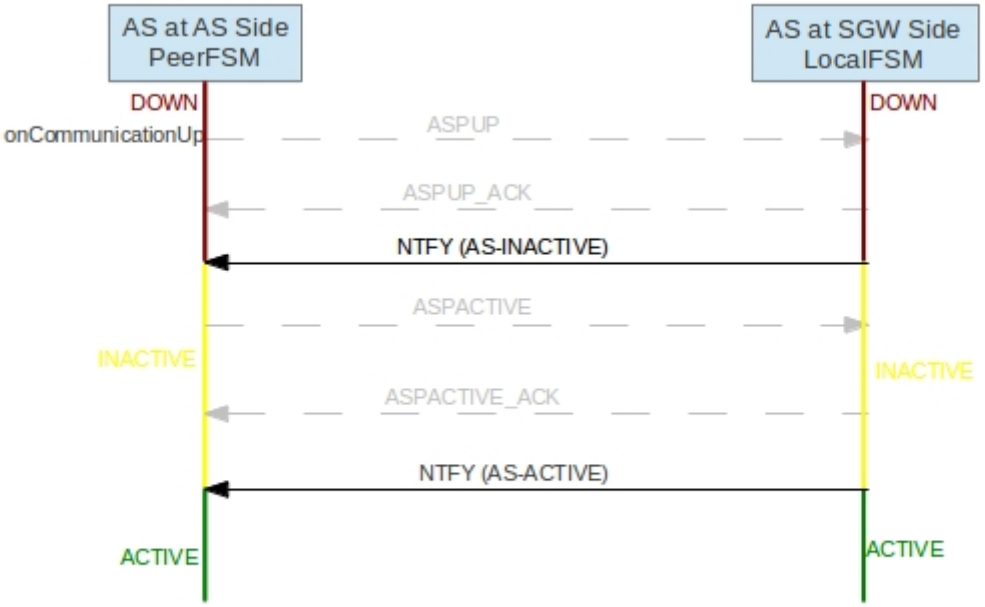


Figure 30. Restcomm M3UA Stack AS state machine for AS - SGW

For IPSP, the ASP state machine remains same as AS-SGW, however for AS the state machine changes a bit as there is no exchange of Notify (NTFY) messages. Figure below shows the various state of AS at IPSP Client and Server side. In below example only Single Exchange of messages are considered for handshake mechanism

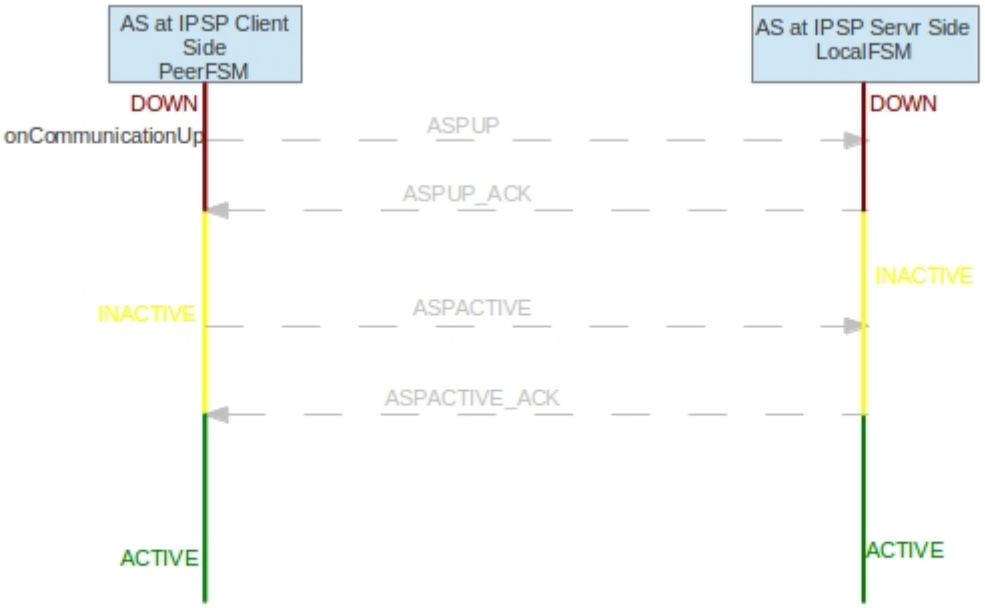


Figure 31. Restcomm M3UA Stack AS state machine for IPSP Client - Server

In Double Exhcnage handshake mechanism both Local and Peer FSM exist in ASP and AS at each side. Hence the number of handshake messages exchanged are twice that of Single Exchange.

Chapter 9. ISUP

(ISDN User Part or ISUP) is part of which is used to establish telephone calls and manage call switches(exchanges). Exchanges are connected via E1 or T1 trunks. Each trunk is divided by means of TDM into time slots. Each time slot is distinguished as circuit. Circuits (identified by code) are used as medium to transmit voice data between user equipment (or exchanges if more than one is involved).

```
allows not only to setup a call, but to exchange information about exchange state and
its resources(circuits).
```

Restcomm is based on Q.76X series of documents.

9.1. ISUP Configuration

Restcomm stack is configured with simple properties. Currently following properties are supported:

Table 6. ISUP Configuration options

Name	Default value	Value range	Description
ni	None, must be provided	0-3	Sets value of network indicator that should be used by stack.
localspc	None, must be provided	0 - (2^14)-1	Sets local signaling point code. It will be used as OPC for outgoing signaling units.
t1	4s	4s - 15s	Sets T1 value. Started when REL is sent. See A.1/Q.764
t5	5 min.	5min - 15 min	Sets T5 value. Started when initial REL is sent. See A.1/Q.764
t7	20s	20s -30s	Sets T7 value. (Re)Started when Address Message is sent. See A.1/Q.764
t2	15s	15s - 60s	Sets T12 value. Started when BLO is sent. See A.1/Q.764

Name	Default value	Value range	Description
t13	5min	5min - 15min	Sets T13 value. Started when initial BLO is sent. See A.1/Q.764
t14	5s	15s - 60s	Sets T14 value. Started when UBL is sent. See A.1/Q.764
t15	5min	5min - 15min	Sets T15 value. Started when initial UBL is sent. See A.1/Q.764
t16	5s	15s - 60s	Sets T16 value. Started when RSC is sent. See A.1/Q.764
t17	5min	5min - 15min	Sets T17 value. Started when initial RSC is sent. See A.1/Q.764
t18	5s	15s - 60s	Sets T18 value. Started when CGB is sent. See A.1/Q.764
t19	5min	5min - 15min	Sets T19 value. Started when initial CGB is sent. See A.1/Q.764
t20	5s	15s - 60s	Sets T20 value. Started when CGU is sent. See A.1/Q.764
t21	5min	5min - 15min	Sets T21 value. Started when initial CGU is sent. See A.1/Q.764
t22	5s	15s - 60s	Sets T22 value. Started when GRS is sent. See A.1/Q.764
t23	5min	5min - 15min	Sets T23 value. Started when initial GRS is sent. See A.1/Q.764
t28	10s	10s	Sets T28 value. Started when CQM is sent. See A.1/Q.764
t33	12s	12s - 15s	Sets T33 value. Started when INR is sent. See A.1/Q.764

Note that before start user must provide two interfaces to stack:

Mtp3UserPart

implementation of transport layer which should be used by stack

CircuitManager

circuit manager implementation. This interface stores information on mapping between `CIC`(Circuit Identification Code) and `DPC`(Destination Point Code) used as destination for outgoing messages.

9.2. ISUP Usage

The `org.mobicents.protocols.ss7.isup.ISUPStack` interface defines the methods required to represent ISUP Protocol Stack. ISUPStack exposes `org.mobicents.protocols.ss7.isup.ISUPProvider`. This interface defines the methods that will be used by any registered ISUP User application implementing the `org.mobicents.protocols.ss7.isup.ISUPListener` to listen ISUP events(messages and timeouts).

9.3. ISUP Example

Below is simple example of stack usage:

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import org.mobicents.protocols.ss7.isup.ISUPEvent;
import org.mobicents.protocols.ss7.isup.ISUPListener;
import org.mobicents.protocols.ss7.isup.ISUPProvider;
import org.mobicents.protocols.ss7.isup.ISUPStack;
import org.mobicents.protocols.ss7.isup.ISUPTimeoutEvent;
import org.mobicents.protocols.ss7.isup.ParameterException;
import org.mobicents.protocols.ss7.isup.impl.ISUPStackImpl;
import org.mobicents.protocols.ss7.isup.message.ISUPMessage;

import org.mobicents.ss7.linkset.oam.Layer4;
import org.mobicents.ss7.linkset.oam.Linkset;

public class ISUPTest implements ISUPListener
{

    protected ISUPStack stack;
    protected ISUPProvider provider;

    protected Linkset isupLinkSet;


    public void setUp() throws Exception {
```

```
        this.isupLinkSet = ....; //same linksets as in SS7Service
        this.stack = new ISUPStackImpl();
        this.stack.configure(getSpecificConfig());
        this.provider = this.stack.getIsupProvider();
        this.provider.addListener(this);
        Mtp3UserPart userPart = // create with proper factory, dahdii, dialogi, m3ua
        this.stack.setMtp3UserPart(userPart);
        CircuitManagerImpl circuitManager = new CircuitManagerImpl();
        circuitManager.addCircuit(1, 431613); // CIC - 1, DPC for it - 431613


        this.stack.setCircuitManager(circuitManager);
        this.stack.start();

    }


    public void onEvent(ISUPEvent event) {
        ISUPMessage msg = event.getMessage();
        switch(msg.getCircuitIdentificationCode().getCIC())
        {
            case AddressCompleteMessage._COMMAND_CODE:
            //only complete
            break;
            case ConnectedMessage._COMMAND_CODE:
            case AnswerMessage._COMMAND_CODE:
            //we are good to go
            ConnectedNumber cn = (ConnectedNumber)msg.getParameter(ConnectedNumber
._PARAMETER_CODE);
            //do something
            break;
            case ReleaseMessage._COMMAND_CODE:
            //remote end does not want to talk
            RealeaseCompleteMessage rlc = provider.getMessageFactory().createRLC();
            rlc.setCircuitIdentificationCode(msg.getCircuitIdentificationCode());
            rlc.setCauseIndicators(   ((ReleaseComplete)msg).getCauseIndicators());
            provider.sendMessage(rlc);

        }

    }

    public void onTimeout(ISUPTimeoutEvent event) {
        switch(event.getTimerId())
        {
            case ISUPTimeoutEvent.T1:
                //do something
                break;
            case ISUPTimeoutEvent.T7:
                //do even more
```

```
                break;
            }

        }

    }
```

Chapter 10. SCCP

The Signaling Connection Control Part (SCCP) is defined in ITU-T Recommendations Q.711-Q.716. SCCP sits on top of Message Transfer Part 3 (MTP3) in the SS7 protocol stack. The SCCP provides additional network layer functions to provide transfer of noncircuit-related (NCR) signaling information, application management procedures and alternative, more flexible methods of routing.

10.1. Routing Management

SCCP provides a routing function that allows signaling messages to be routed to a signaling point based on dialed digits, for example. This capability is known as Global Title Translation (GTT), which translates what is known as a global title (for example, dialed digits for a toll free number) into a signaling point code and a subsystem number so that it can be processed at the correct application.

10.1.1. GTT Configuration

GTT is performed in two stages. First is matching the rule and second is actual translation.

For matching the rule, the called party address global title digits are matched with <digits> configured while defining the Rule. Once the digits match actual translation is done. Also for rule matching GT Indicator, translation type, numbering plan and nature-of-address of a rule and of message called party address must be equal.

Matching rule

As explained previously, the <digits> can be divided into sections using the "/" separate character. Each section defines set of digits to be matched. Wild card * can be used to match any digits and ? can be used to match exatcly one digit. For example Rule is to match starting 4 digits (should be 1234) and doesn’t care for rest; the <digits> in the command will be 1234/*. If the Rule is such that starting 3 digits should be 123, doesn’t care for other three digits but last two digits should be 78; the <digits> in the command will be 123/???/78. If digit to digit matching is needed the the <digits> in the command will be exact digits to be matched without sections.

Translation

Resulting Called Party Address will contain:

- PC in the generated CalledPartyAddress is taken from primary/backup address. This PC will not be encoded into a message if the following option is set: sccp set removespc true.
- SSN will be taken from primary/backup address. If it is absent there - will be taken from an original message address. If also absent - SSN will not be included.
- GT type (0001, 0010, 0011, 0100) is taken from from AI (AddressIndicator) of primary/backup address. If AI of primary/backup address contains "no GT" (or bad value) but after address translating we need GT - it will be taken from AI of an original message address.
- NAI, TT, NP (and Encoding schema) is taken from GT of primary/backup address. If no GT in primary/backup address but after address translating we need GT - it will be taken from GT

of an original message address.

- Digits for CalledPartyAddress will be generated in following way. For translation each section in <mask> defined while creating the Rule, defines how replacement operation is performed. If <mask> defines K (KEEP), the originally dialed digits are kept and if <mask> defines R (REPLACE) the digits from primary address or back address are used. The primary/backup address should always define the point code and the translated address will always have this point code. If the primary/backup address defines the subsystem number the translated address will also have this subsystem number. The address-indicator of translated address is always from primary/backup address. See below examples

1. Example 1 : Remove the Global Title and add PC and SSN

Element	Address Indicator	PC	SSN	TT	NP	NAI	Digits
Dialed Address	0 0 0 1 0 0 0 0			1			123456789
Rule Address	0 0 0 1 0 0 0 0			1			123456789
Rule mask							R
Primary Address	0 1 0 0 0 0 1 1	123	8				-
Translated Address	0 1 0 0 0 0 1 1	123	8				

Figure 32. GTT - Example 1

2. Example 2 : Partial Match Match a eight digit number starting "800", followed by any four digits, then "9". If the translated digits is not null and if the primary/backup address has no Global Title, the Global Title from dialed address is kept with new translated digits.

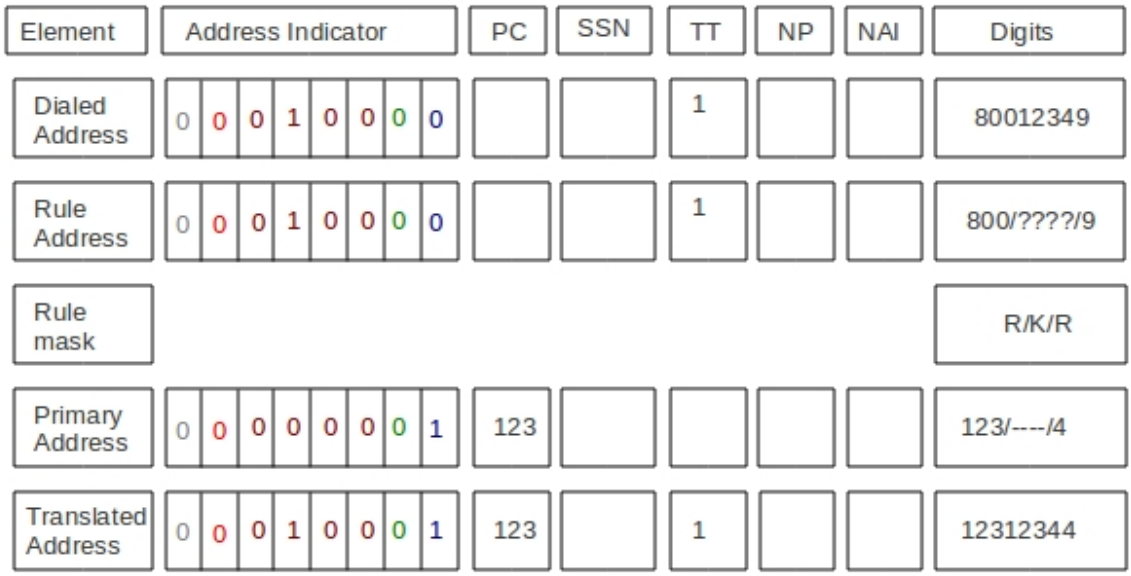


Figure 33. GTT - Example 2

3. Example 3 : Partial Match Match "800800", followed by any digits Remove the first six digits. Keep any following digits in the Input. Add a PC(123) and SSN (8).

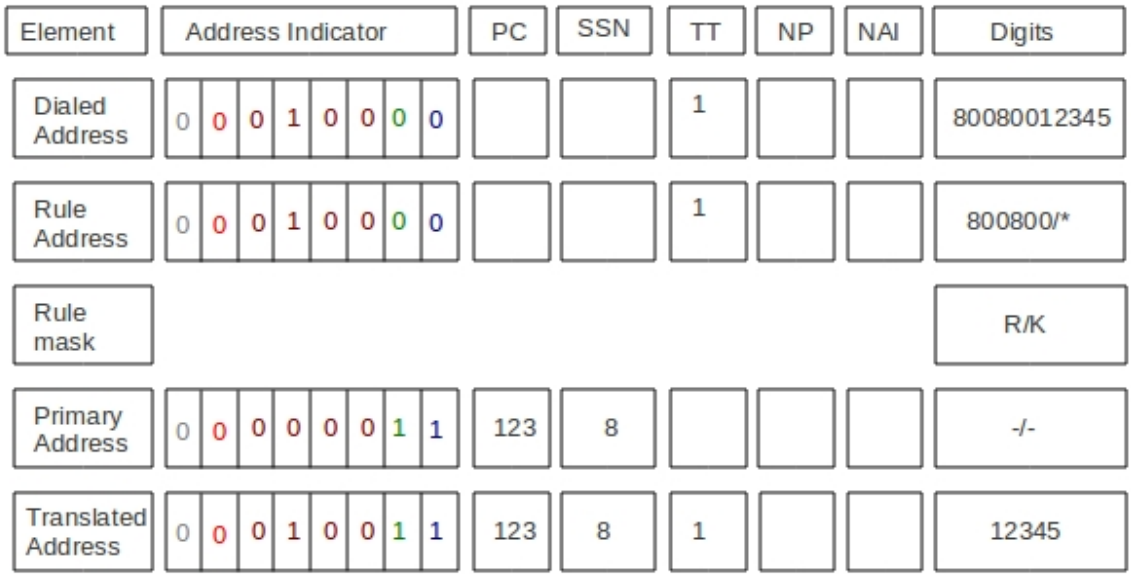


Figure 34. GTT - Example 3

4. Example 4 : Partial Match Match any digits keep the digits in the and add a PC(123) and SSN (8). If the translated digits is not null and if the primary/backup address has no Global Title, the Global Title from dialed address is kept with new translated digits.

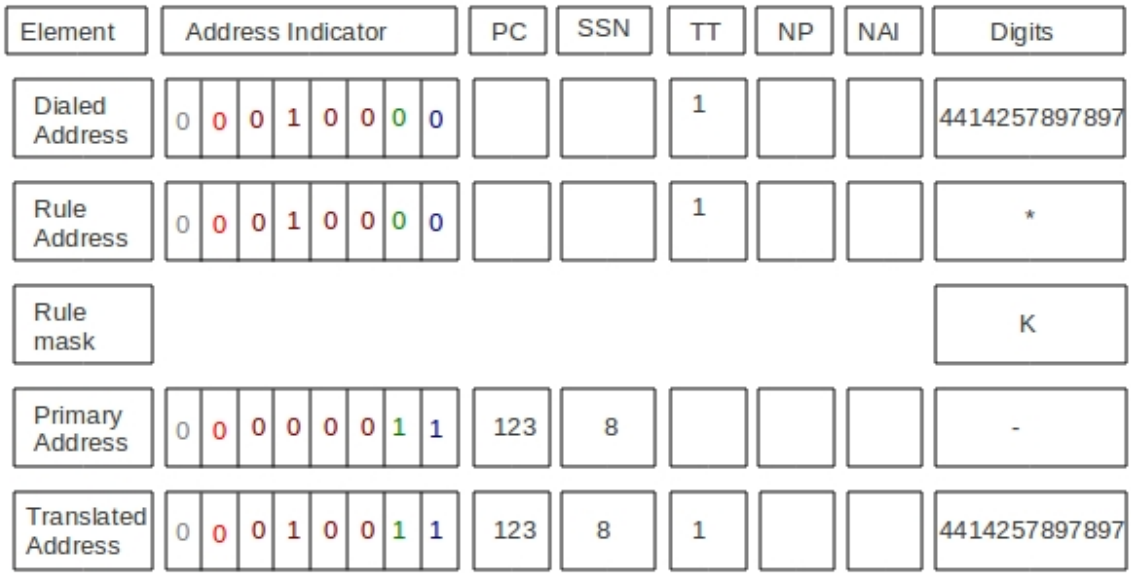


Figure 35. GTT - Example 4

10.2. SCCP Usage

The instance of `org.mobicents.protocols.ss7.sccp.SccpStack` acts as starting point. All the sccp messages sent by SCCP User Part are routed as per the rule configured in Router

The term SCCP User Part refers to the applications that use SCCP's services.

The SCCP User Part gets handle to `SccpStack` by doing a JNDI look-up.

`SccpStack` exposes `org.mobicents.protocols.ss7.sccp.SccpProvider` that interacts directly with SccpStack. This interface defines the methods that will be used by SCCP User Part to send `org.mobicents.protocols.ss7.sccp.message.SccpMessage` and register [class]`org.mobicents.protocols.ss7.sccp.SccpListener's to listen for incoming SCCP messages.

SCCP User Part registers SccpListener for specific local subsystem number. For every incoming SccpMessage, if the called subsystem matches with this local subsystem, the corresponding SccpListner is called.

SccpProvider also exposes `org.mobicents.protocols.ss7.sccp.message.MessageFactory` and `org.mobicents.protocols.ss7.sccp.parameter.ParameterFactory` to create new SccpMessage. For transfer data via connectionless service `org.mobicents.protocols.ss7.sccp.message.SccpDataMessage` is used. (This class use UDT, XUDT, LUDT SCCP message type for message transfer.)

10.3. SCCP User Part Example

Below is SCCP User Part example listening for incoming SCCP message and sending back new message

```
package org.mobicents.protocols.ss7.sccp.impl;
```

```
import java.io.IOException;

import org.mobicents.protocols.ss7.indicator.NatureOfAddress;
import org.mobicents.protocols.ss7.indicator.NumberingPlan;
import org.mobicents.protocols.ss7.indicator.RoutingIndicator;
import org.mobicents.protocols.ss7.sccp.RemoteSccpStatus;
import org.mobicents.protocols.ss7.sccp.SccpListener;
import org.mobicents.protocols.ss7.sccp.SccpProvider;
import org.mobicents.protocols.ss7.sccp.SignallingPointStatus;
import org.mobicents.protocols.ss7.sccp.message.SccpDataMessage;
import org.mobicents.protocols.ss7.sccp.message.SccpNoticeMessage;
import org.mobicents.protocols.ss7.sccp.parameter.GlobalTitle;
import org.mobicents.protocols.ss7.sccp.parameter.HopCounter;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;

public class Test implements SccpListener {
        private SccpProvider sccpProvider;
        private SccpAddress localAddress;
        private int localSsn = 8;

        private static SccpProvider getSccpProvider() {
                Mtp3UserPartImpl mtp3UserPart1 = null;
                // ......
                // ......
                SccpStackImpl sccpStack1 = new SccpStackImpl("testSccpStack");
                sccpStack1.setMtp3UserPart(1, mtp3UserPart1);
                sccpStack1.start();
                return sccpStack1.getSccpProvider();
        }

        public void start() throws Exception {
                this.sccpProvider = getSccpProvider();
                int translationType = 0;
                GlobalTitle gt = GlobalTitle.getInstance(translationType,
                                NumberingPlan.ISDN_MOBILE, NatureOfAddress.NATIONAL,
"1234");
                localAddress = new SccpAddress(RoutingIndicator
.ROUTING_BASED_ON_GLOBAL_TITLE, -1, gt, 0);
                this.sccpProvider.registerSccpListener(this.localSsn, this);
        }

        public void stop() {
                this.sccpProvider.deregisterSccpListener(this.localSsn);
        }

        @Override
        public void onMessage(SccpDataMessage message) {
                localAddress = message.getCalledPartyAddress();
                SccpAddress remoteAddress = message.getCallingPartyAddress();
                // now decode content
```

```
                byte[] data = message.getData();
                // processing a request
                byte[] answerData = new byte[10];
                // put custom executing code here and fill answerData
                HopCounter hc = this.sccpProvider.getParameterFactory
().createHopCounter(5);
                SccpDataMessage sccpAnswer = this.sccpProvider.getMessageFactory
().createDataMessageClass1(
                                remoteAddress, localAddress, answerData, message
.getSls(),
                                localSsn, false, hc, null);
                try {
                        this.sccpProvider.send(sccpAnswer);
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        @Override
        public void onNotice(SccpNoticeMessage message) {
        }

        public void onCoordRequest(int dpc, int ssn, int multiplicityIndicator) {
        }

        public void onCoordResponse(int dpc, int ssn, int multiplicityIndicator) {
        }

        public void onState(int dpc, int ssn, boolean inService, int
multiplicityIndicator) {
        }

        @Override
        public void onPcState(int dpc, SignallingPointStatus status, int
restrictedImportanceLevel,
                        RemoteSccpStatus remoteSccpStatus) {
        }

}
```

Chapter 11. TCAP

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.

11.1. Restcomm jSS7 TCAP Usage

The `org.mobicents.protocols.ss7.tcap.api.TCAPStack` interface defines the methods required to represent the TCAP Protocol Stack. TCAPStack exposes `org.mobicents.protocols.ss7.tcap.api.TCAPProvider` that interacts directly with the TCAPStack. TCAPProvider defines methods that will be used by TCAP User Part to create new `org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog` to be sent across the network. TCAP User Part also allows to register`org.mobicents.protocols.ss7.tcap.api.TCListener` to listen for TCAP messages.

TCAPProvider also exposes `org.mobicents.protocols.ss7.tcap.api.DialogPrimitiveFactory` to create dialog primitives and `org.mobicents.protocols.ss7.tcap.api.ComponentPrimitiveFactory` to create components. Components are a means of invoking an operation at a remote node.

The UML Class Diagram is depicted in the figure below:

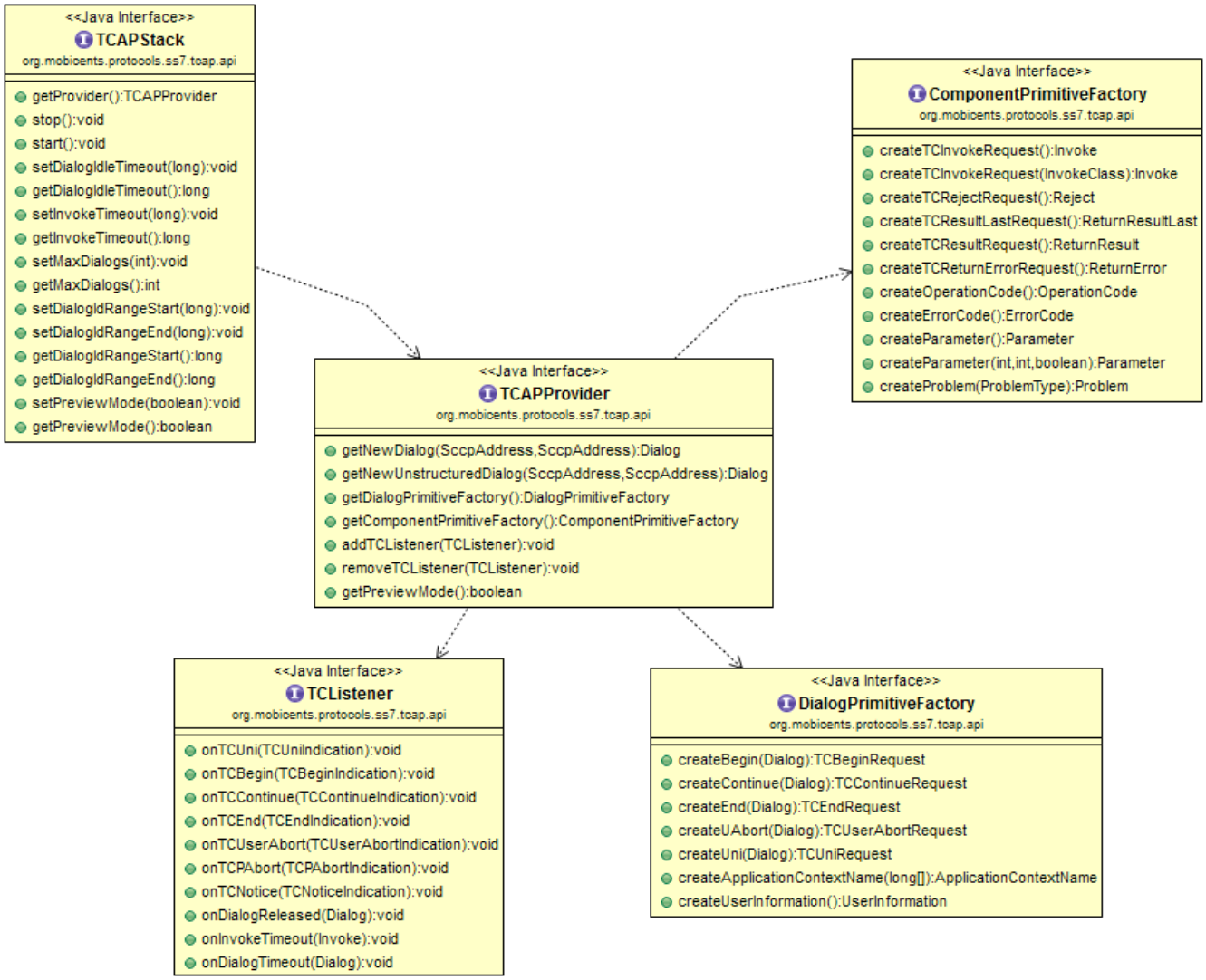


Figure 36. Restcomm jSS7 Stack TCAP Class Diagram

The `org.mobicents.protocols.ss7.tcap.TCAPStackImpl` is a concrete implementation of `TCAPStack`. The TCAP User Part gets access to `TCAPProvider` by doing JNDI lookup as explained in the SS7 Service.

```
        InitialContext ctx = new InitialContext();
        try {
            String providerJndiName = "java:/restcomm/ss7/tcap";
            this.tcapProvider = ((TCAPProvider) ctx.lookup(providerJndiName));
        } finally {
            ctx.close();
        }
```

The TCAP User Part should register the concrete implementation of `TCListener` with `TCAPProvider` to listen for incoming TCAP messages.

```
    public class ClientTest implements TCListener{
            .....
            tcapProvider.addTCListener(this);
            ....
    }
```

TCAP User Part leverages `TCAPProvider` to create a new Dialog. The components between the nodes are exchanged within this Dialog.

```
        SccpAddress localAddress = new SccpAddress(RoutingIndicator
.ROUTING_BASED_ON_DPC_AND_SSN, 1, null, 8);
        SccpAddress remoteAddress = new SccpAddress(RoutingIndicator
.ROUTING_BASED_ON_DPC_AND_SSN, 2, null, 8);
        clientDialog = this.tcapProvider.getNewDialog(localAddress, remoteAddress);
```

The TCAP User Part leverages `ComponentPrimitiveFactory` to create new components. These components are sent using the dialog.

Below is a list of common scenarios using the TCAP stack :

- Creating a TCAP Dialog by invoking the methods `TCAPProvider.getNewDialog()` or `getNewUnstructuredDialog()`
- Adding components into a Dialog for sending by `Dialog.sendComponent();`
- Sending a TCAP message TC-UNI, TC-BEGIN, TC-CONTINUE, TC-END or TC-ABORT via `Dialog.send()` methods.
- Waiting for responses from a peer
- When the TCAP stack receives a message from a peer, events like `TCListener.onTCUni()`, `onTCBegin()`, `onTCContinue()`, `onTCEnd()`, `onTCUserAbort()`, `onTCPAbort()` will be invoked.
- After an Invoke component is received, a TCAP-User should process it and do one of the below:
  - send a response (ReturnResult, ReturnResulLast components) or
  - send an error (ReturnError or Reject components) or
  - invoke Dialog.processInvokeWithoutAnswer() method if TCAP-Users will not answer to the Invoke.

```
        //create some INVOKE
        Invoke invoke = cpFactory.createTCInvokeRequest();
        invoke.setInvokeId(this.clientDialog.getNewInvokeId());
        OperationCode oc = cpFactory.createOperationCode();
        oc.setLocalOperationCode(12L);
        invoke.setOperationCode(oc);
        //no parameter
        this.clientDialog.sendComponent(invoke);
```

11.2. Restcomm jSS7 TCAP User Part Example

Below is a TCAP User Part example. This example creates a dialog and exchanges messages within a structured dialog. Refer to source for function calls.

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

```
import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.mobicents.protocols.ss7.indicator.RoutingIndicator;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;
import org.mobicents.protocols.ss7.tcap.api.ComponentPrimitiveFactory;
import org.mobicents.protocols.ss7.tcap.api.TCAPException;
import org.mobicents.protocols.ss7.tcap.api.TCAPProvider;
import org.mobicents.protocols.ss7.tcap.api.TCAPSendException;
import org.mobicents.protocols.ss7.tcap.api.TCListener;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCBeginIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCBeginRequest;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCContinueIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCEndIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCEndRequest;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCNoticeIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCPAbortIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCUniIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TCUserAbortIndication;
import org.mobicents.protocols.ss7.tcap.api.tc.dialog.events.TerminationType;
import org.mobicents.protocols.ss7.tcap.asn.ApplicationContextName;
import org.mobicents.protocols.ss7.tcap.asn.comp.Invoke;
import org.mobicents.protocols.ss7.tcap.asn.comp.OperationCode;

/**
 * Simple example demonstrates how to use TCAP Stack
 *
 * @author Amit Bhayani
 *
 */
public class ClientTest implements TCListener {
    // encoded Application Context Name
    public static final long[] _ACN_ = new long[] { 0, 4, 0, 0, 1, 0, 19, 2 };
    private TCAPProvider tcapProvider;
    private Dialog clientDialog;

    ClientTest() throws NamingException {

        InitialContext ctx = new InitialContext();
        try {
            String providerJndiName = "java:/restcomm/ss7/tcap";
            this.tcapProvider = ((TCAPProvider) ctx.lookup(providerJndiName));
        } finally {
            ctx.close();
        }

        this.tcapProvider.addTCListener(this);
    }
```

```
    public void sendInvoke() throws TCAPException, TCAPSendException {
        SccpAddress localAddress = new SccpAddress(RoutingIndicator
.ROUTING_BASED_ON_DPC_AND_SSN, 1, null, 8);
        SccpAddress remoteAddress = new SccpAddress(RoutingIndicator
.ROUTING_BASED_ON_DPC_AND_SSN, 2, null, 8);

        clientDialog = this.tcapProvider.getNewDialog(localAddress, remoteAddress);
        ComponentPrimitiveFactory cpFactory = this.tcapProvider
.getComponentPrimitiveFactory();

        // create some INVOKE
        Invoke invoke = cpFactory.createTCInvokeRequest();
        invoke.setInvokeId(this.clientDialog.getNewInvokeId());
        OperationCode oc = cpFactory.createOperationCode();
        oc.setLocalOperationCode(12L);
        invoke.setOperationCode(oc);
        // no parameter
        this.clientDialog.sendComponent(invoke);
        ApplicationContextName acn = this.tcapProvider.getDialogPrimitiveFactory
().createApplicationContextName(_ACN_);
        // UI is optional!
        TCBeginRequest tcbr = this.tcapProvider.getDialogPrimitiveFactory
().createBegin(this.clientDialog);
        tcbr.setApplicationContextName(acn);
        this.clientDialog.send(tcbr);
    }

    public void onDialogReleased(Dialog d) {
    }

    public void onInvokeTimeout(Invoke tcInvokeRequest) {
    }

    public void onDialogTimeout(Dialog d) {
        d.keepAlive();
    }

    public void onTCBegin(TCBeginIndication ind) {
    }

    public void onTCContinue(TCContinueIndication ind) {
        // send end
        TCEndRequest end = this.tcapProvider.getDialogPrimitiveFactory().createEnd(
ind.getDialog());
        end.setTermination(TerminationType.Basic);
        try {
            ind.getDialog().send(end);
        } catch (TCAPSendException e) {
            throw new RuntimeException(e);
        }
    }
```

```
    public void onTCEnd(TCEndIndication ind) {
        // should not happen, in this scenario, we send data.
    }

    public void onTCUni(TCUniIndication ind) {
        // not going to happen
    }

    public void onTCPAbort(TCPAbortIndication ind) {
        // TODO Auto-generated method stub
    }

    public void onTCUserAbort(TCUserAbortIndication ind) {
        // TODO Auto-generated method stub
    }

    public void onTCNotice(TCNoticeIndication ind) {
        // TODO Auto-generated method stub

    }

    public static void main(String[] args) {

        try {
            ClientTest c = new ClientTest();
            c.sendInvoke();
        } catch (NamingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (TCAPException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (TCAPSendException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

11.3. Restcomm jSS7 TCAP statistic counters

Below is a list of provided by TCAP Stack statistic counters. Please see the info how to enable statistics in the following chapters: Statistics Enabled, Statistics.

Table 7. TCAP statistic counters

CounterType	Id	Description
Summary	TcUniReceivedCount	A count of received TC-UNI messages
Summary	TcUniSentCount	A count of sent TC-UNI messages
Summary	TcBeginReceivedCount	A count of received TC-BEGIN messages
Summary	TcBeginSentCount	A count of sent TC-BEGIN messages
Summary	TcContinueReceivedCount	A count of received TC-CONTINUE messages
Summary	TcContinueSentCount	A count of sent TC-CONTINUE messages
Summary	TcEndReceivedCount	A count of received TC-END messages
Summary	TcEndSentCount	A count of sent TC-END messages
Summary	TcPAbortReceivedCount	A count of received TC-PROVIDER-ABORT messages
Summary	TcPAbortSentCount	A count of sent TC-PROVIDER-ABORT messages
Summary	TcUserAbortReceivedCount	A count of received TC-USER-ABORT messages
Summary	TcUserAbortSentCount	A count of sent TC-USER-ABORT messages
Summary	InvokeReceivedCount	A count of received Invoke components
Summary	InvokeSentCount	A count of sent Invoke components
Summary	ReturnResultReceivedCount	A count of received ReturtResult components
Summary	ReturnResultSentCount	A count of sent ReturtResult components
Summary	ReturnResultLastReceivedCount	A count of received ReturtResultLast components
Summary	ReturnResultLastSentCount	A count of sent ReturtResultLast components
Summary	ReturnErrorReceivedCount	A count of received ReturnError components

CounterType	Id	Description
Summary	ReturnErrorSentCount	A count of sent ReturnError components
Summary	RejectReceivedCount	A count of received Reject components
Summary	RejectSentCount	A count of sent Reject components
Summary	DialogTimeoutCount	A count of received DialogTimeouts
Summary	DialogReleaseCount	A count of received DialogReleases
Summary	AllEstablishedDialogsCount	A count of all established Dialogs
Summary	AllLocalEstablishedDialogsCount	A count of all established local originated Dialogs
Summary	AllRemoteEstablishedDialogsCount	A count of all established remote originated Dialogs
Minimal	MinDialogsCount	A min count of established Dialogs
Maximal	MaxDialogsCount	A max count of established Dialogs
SummaryDouble	AllDialogsDuration	A total duration of all released Dialogs (in seconds)
Average	AverageDialogsDuration	An average duration of all released Dialogs (in seconds)
ComplexValue	OutgoingDialogsPerApplicatioContextName	An outgoing Dialogs count per ApplicationContextNames (in string form)
ComplexValue	IncomingDialogsPerApplicatioContextName	An incoming Dialogs count per ApplicationContextNames (in string form)
ComplexValue	OutgoingInvokesPerOperationCode	An outgoing Invokes count per OperationCodes
ComplexValue	IncomingInvokesPerOperationCode	An incoming Invokes count per OperationCodes
ComplexValue	OutgoingErrorsPerErrorCode	An outgoing ReturtError count per ErrorCodes

CounterType	Id	Description
ComplexValue	IncomingErrorsPerErrorCode	An incoming ReturtError count per ErrorCodes
ComplexValue	OutgoingRejectPerProblem	An outgoing Reject count per Problem
ComplexValue	IncomingRejectPerProblem	An incoming Reject count per Problem

Chapter 12. MAP

Mobile application part (MAP) is the protocol that is used to allow the network nodes within the Network Switching Subsystem (NSS) to communicate with each other in order 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 network. This application layer provides a standardized set of services. uses the services of the GSM network, specifically the Signaling Connection Control Part (SCCP) and the Transaction Capabilities Application Part (TCAP).

For better understanding of this chapter you must be familiar with the specifications defined in `GSM 09.02`.

MAP has full implementation for USSD, SMS and LMS (Location Management Service) services and partial implementation for Mobility Services (updateLocation, sendAuthenticationInfo, anyTimeInterrogation, checkIMEI operations). You can find the list of implemented MAP messages here: MAPMessagesImplemented.ods. The document uses color coding to represent the status of the implementation:

Green

Fully Implemented

Red

Interface available (but not yet implemented)

Yellow

Implementation in progress

Restcomm jSS7 Any contribution to implement other messages are welcome. We will provide you with all the help that you may require initially.

12.1. jSS7 MAP

The methods required to represent the MAP Protocol Stack are defined in the interface `org.mobicents.protocols.ss7.map.api.MAPStack` which exposes `org.mobicents.protocols.ss7.map.api.MAPProvider`that interacts directly with the `MAPStack`. This interface defines the methods that will be used by any registered MAP-User application.

A MAP-User application must implement interfaces to listen for MAP messages and also implement interfaces for dialogue and component handling primitives. One such interface is `org.mobicents.protocols.ss7.map.api.MAPDialogListener` which must be implemented for listening associated with dialog events. Others are one or more interfaces for listening associated with component events based on `org.mobicents.protocols.ss7.map.api.MAPServiceListener` interface.

Below is a list of MAP Service Listener classes. A MAP-User interested in listening for messages

specific to a particular MAP Service must implement the corresponding MAPServiceListener.

LMS operations

`org.mobicents.protocols.ss7.map.api.service.lsm.MAPServiceLsmListener`

Mobility operations

`org.mobicents.protocols.ss7.map.api.service.mobility.MAPServiceMobilityListener`

OAM operations

`org.mobicents.protocols.ss7.map.api.service.oam.MAPServiceOamListener`

PDP context activation operations

`org.mobicents.protocols.ss7.map.api.service.pdpContextActivation.MAPServicePdpContextActivationListener`

SMS operations

`org.mobicents.protocols.ss7.map.api.service.sms.MAPServiceSmsListener`

Supplementary operations

`org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListener`

A MAP-User interested in all the services must implement all the above Service Listener classes.

The `org.mobicents.protocols.ss7.map.MAPStackImpl` class is a concrete implementation of `MAPStack`. The MAP-User application gets a reference to `MAPProvider` by doing a JNDI lookup as explained in SS7 Service.

```
String providerJndiName = "java:/restcomm/ss7/map";
this.mapProvider = ((MAPProvider) ctx.lookup(providerJndiName));
...
```

To listen for incoming MAP Dialog and MAP Primitive messages, the MAP-User application should register the concrete implementation of `MAPDialogListener` with `MAPProvider`.

To listen for incoming MAP Service messages, the MAP-User application should register the concrete implementation of `MAPServiceListener` with the corresponding `MAPServiceBase` `. The following [class]`MAPServiceBase` services are exposed by the `MAPProvider`:

Call handling service

`org.mobicents.protocols.ss7.map.api.service.callhandling.MAPServiceCallHandling`

LSM service

`org.mobicents.protocols.ss7.map.api.service.lsm.MAPServiceLsm`

Mobility service

`org.mobicents.protocols.ss7.map.api.service.mobility.MAPServiceMobility`

OAM service

`org.mobicents.protocols.ss7.map.api.service.oam.MAPServiceOam`

PDP context activation service

`org.mobicents.protocols.ss7.map.api.service.pdpContextActivation.MAPServicePdpContextActivation`

SMS service

`org.mobicents.protocols.ss7.map.api.service.sms.MAPServiceSms`

Supplementary service

`org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementary`

```
public class UssdClientExample implements MAPDialogListener,
MAPServiceSupplementaryListener {
        ....
        mapProvider.addMAPDialogListener(this);
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
        ....
}
```

Prior to using any MAP specific service, the corresponding service should be activated as shown below:

```
        ....
        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
        ....
```

The MAP-User Application leverages:

- `MAPParameterFactory` to create instances of `USSDString`, `ISDNAddressString`and many other primitives that are used by MAP services.
- `MAPSmsTpduParameterFactory` to create instances of SMS TPDU primitives used for sending SMS messages like `SmsDeliverTpdu` or `SmsSubmitTpdu`.
- `MAPErrorMessageFactory` to create instances of MAP error messages like `MAPErrorMessageSystemFailure`.

```
        MapParameterFactory paramFact = mapProvider.getMapServiceFactory();
        USSDString ussdString = paramFact.createUSSDString("*125*+31628839999#",
                null);
        ISDNAddressString msisdn = paramFact.createISDNAddressString(
                AddressNature.international_number, NumberingPlan.ISDN,
                "31628838002");
```

12.2. jSS7 Sending a MAP request (processUnstructuredSS-Request as an example)

For sending a MAP request, you must do the following at the client side:

- Create a new MAP Dialog

```
// First create Dialog
SccpAddress origAddress = createLocalAddress();
ISDNAddressString origReference = client.getMAPProvider().getMAPParameterFactory().
    createISDNAddressString(AddressNature.international_number, NumberingPlan
.land_mobile, "31628968300");
SccpAddress destAddress = createRemoteAddress();
ISDNAddressString destReference = client.getMAPProvider().getMAPParameterFactory().
    createISDNAddressString(AddressNature.international_number, NumberingPlan
.land_mobile, "204208300008002");

currentMapDialog = mapProvider.getMAPServiceSupplementary().
    createNewDialog(MAPApplicationContext.getInstance(MAPApplicationContextName
.networkUnstructuredSsContext,
    MAPApplicationContextVersion.version2), origAddress,
    destReference, remoteAddress, destReference);
```

- Add an Invoke component (processUnstructuredSS-Request message)

```
// The dataCodingScheme is still byte, as I am not exactly getting how
// to encode/decode this.
byte ussdDataCodingScheme = 0x0f;
// The Charset is null, here we let system use default Charset (UTF-7 as
// explained in GSM 03.38. However if MAP-User wants, it can set its own
// impl of Charset
USSDString ussdString = paramFact.createUSSDString(ussdMessage, null);
ISDNAddressString msisdn = client.getMAPProvider().getMAPParameterFactory().
createISDNAddressString(AddressNature.international_number, NumberingPlan.ISDN,
"31628838002");
currentMapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme, ussdString,
alertingPattern, msisdn);
```

- Send a TC-Begin message to the server peer

```
currentMapDialog.send();
```

- Wait for a response from the server

At the server side, when the TC-Begin message is received, the following sequence of events occur:

```
void MAPDialogListener. onDialogRequest(MAPDialog mapDialog, AddressString
destReference,
    AddressString origReference, MAPExtensionContainer extensionContainer);
```

This is the request for MAP Dialog processing. A MAP-User can reject the Dialog by invoking the `mapDialog.refuse()` method.

This is followed by the events (one or more) corresponding to the incoming primitives. In this case it is:

```
void MAPServiceSupplementaryListener.onProcessUnstructuredSSRequest
(ProcessUnstructuredSSRequest procUnstrReqInd);
```

When processing component-dependant messages, you can add response components. In this case it is processUnstructuredSS-Response as an example:

```
        USSDString ussdString = ind.getUSSDString();
        String request = ussdString.getString();

        // processing USSD request
        String response = "Your balans is 100$";

        // The dataCodingScheme is still byte, as I am not exactly getting how
        // to encode/decode this.
        byte ussdDataCodingScheme = 0x0f;
        USSDString ussdResponse = paramFact.createUSSDString(response, null);

        try {mapDialog.addProcessUnstructuredSSResponse(ind.getInvokeId(),
ussdDataCodingScheme, ussdResponse);
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
```

If preparing the response takes more time, you should return the control and prepare the answer asynchronously in a separate thread.

If error or reject primitives are included in a TCAP message, the following events occur:

```
public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage);
public void onProviderErrorComponent(MAPDialog mapDialog, Long invokeId,
MAPProviderError providerError);
public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem);
```

After all incoming components have been processed, the event `onDialogDelimiter(MAPDialog mapDialog);` or in the case of TC-END, `onDialogClose(MAPDialog mapDialog)` is invoked. If all response

components have been prepared you can tell the stack to send the response:

- `mapDialog.close(false);` - to send TC-END
- `mapDialog.send();` - to send TC-CONTINUE
- `mapDialog.close(true);` - sends TC-END without any components (prearrangedEnd)

Instead of `send()` and `close(boolean prearrangedEnd)` methods you can invoke `sendDelayed()` or `closeDelayed(boolean prearrangedEnd)` methods. These methods are similar to `send()` and `close()` methods, but when these methods are invoked from MAP Service message handlers (component handler methods called by stack while parsing incoming components), real sending and dialog closing will occur only when all incoming component events and `onDialogDelimiter()` ` or `onDialogClose()` is processed. If all response components have been prepared you should return the control and send a response when all components are ready.

In case of an error, you can terminate a MAP dialog in any method by invoking

- `mapDialog.abort(mapUserAbortChoice);` - sends TC-U-ABORT primitive

If there are no local actions or there is no response from a peer for a long time, a timeout occurs and the following methods are invoked:

- `MAPDialogListener.onDialogTimeout(MAPDialog mapDialog);`
- `MAPServiceListener.onInvokeTimeout(MAPDialog mapDialog, Long invokeId);`

In the `onDialogTimeout()` method you can invoke `mapDialog.keepAlive();` to prevent a Dialog from closing. For preventing an Invoke timeout you should invoke `resetInvokeTimer(Long invokeId);` before `onInvokeTimeout()` occurs.

12.3. jSS7 MAP Usage

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

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.mobicents.protocols.ss7.map.api.MAPApplicationContext;
import org.mobicents.protocols.ss7.map.api.MAPApplicationContextName;
import org.mobicents.protocols.ss7.map.api.MAPApplicationContextVersion;
import org.mobicents.protocols.ss7.map.api.MAPDialog;
import org.mobicents.protocols.ss7.map.api.MAPDialogListener;
import org.mobicents.protocols.ss7.map.api.MAPException;
import org.mobicents.protocols.ss7.map.api.MAPMessage;
import org.mobicents.protocols.ss7.map.api.MAPParameterFactory;
import org.mobicents.protocols.ss7.map.api.MAPProvider;
import org.mobicents.protocols.ss7.map.api.datacoding.CBSDataCodingScheme;
import org.mobicents.protocols.ss7.map.api.dialog.MAPAbortProviderReason;
import org.mobicents.protocols.ss7.map.api.dialog.MAPAbortSource;
import org.mobicents.protocols.ss7.map.api.dialog.MAPNoticeProblemDiagnostic;
import org.mobicents.protocols.ss7.map.api.dialog.MAPRefuseReason;
```

```
import org.mobicents.protocols.ss7.map.api.dialog.MAPUserAbortChoice;
import org.mobicents.protocols.ss7.map.api.errors.MAPErrorMessage;
import org.mobicents.protocols.ss7.map.api.primitives.AddressString;
import org.mobicents.protocols.ss7.map.api.primitives.AlertingPattern;
import org.mobicents.protocols.ss7.map.api.primitives.IMSI;
import org.mobicents.protocols.ss7.map.api.primitives.ISDNAddressString;
import org.mobicents.protocols.ss7.map.api.primitives.MAPExtensionContainer;
import org.mobicents.protocols.ss7.map.api.primitives.USSDString;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.MAPDialogSupplementary;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListe
ner;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRequest
;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRespons
e;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyRequest;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyResponse
;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSRequest;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSResponse;
import org.mobicents.protocols.ss7.map.datacoding.CBSDataCodingSchemeImpl;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;
import org.mobicents.protocols.ss7.tcap.asn.ApplicationContextName;
import org.mobicents.protocols.ss7.tcap.asn.comp.Problem;

/**
 * A simple example show-casing how to use MAP stack. Demonstrates how new MAP
 * Dialog is craeted and Invoke is sent to peer.
 *
 * @author Amit Bhayani
 *
 */
public class UssdClientExample implements MAPDialogListener,
MAPServiceSupplementaryListener {

    private MAPProvider mapProvider;
    private MAPParameterFactory paramFact;

    public UssdClientExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {String providerJndiName = "java:/restcomm/ss7/map";this.mapProvider =
((MAPProvider) ctx.lookup(providerJndiName));
        } finally {ctx.close();
```

```
        }
    }

    public MAPProvider getMAPProvider() {
        return mapProvider;
    }

    public void start() {
        // Listen for Dialog events
        mapProvider.addMAPDialogListener(this);
        // Listen for USSD related messages
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);

        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
    }

    public void stop() {
        mapProvider.getMAPServiceSupplementary().deactivate();
    }

    public void sendProcessUssdRequest(SccpAddress origAddress, AddressString
origReference, SccpAddress remoteAddress,AddressString destReference, String
ussdMessage, AlertingPattern alertingPattern, ISDNAddressString msisdn)throws
MAPException {
        // First create Dialog
        MAPDialogSupplementary currentMapDialog = mapProvider
.getMAPServiceSupplementary().createNewDialog( MAPApplicationContext.getInstance
(MAPApplicationContextName.networkUnstructuredSsContext,
MAPApplicationContextVersion.version2), origAddress, destReference, remoteAddress,
destReference);

        CBSDataCodingScheme ussdDataCodingScheme = new CBSDataCodingSchemeImpl(0x0f);
        // The Charset is null, here we let system use default Charset (UTF-7 as
        // explained in GSM 03.38. However if MAP-User wants, it can set its own
        // impl of Charset
        USSDString ussdString = paramFact.createUSSDString(ussdMessage, null, null);

        currentMapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme,
ussdString, alertingPattern, msisdn);
        // This will initiate the TC-BEGIN with INVOKE component
        currentMapDialog.send();
    }

    public void onProcessUnstructuredSSResponse(ProcessUnstructuredSSResponse ind) {
        USSDString ussdString = ind.getUSSDString();
        try {String response = ussdString.getString(null);// processing USSD response
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
    }
```

```
    public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage) {
        // TODO Auto-generated method stub

    }

    public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem,
boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    public void onInvokeTimeout(MAPDialog mapDialog, Long invokeId) {
        // TODO Auto-generated method stub

    }

    public void onMAPMessage(MAPMessage mapMessage) {
        // TODO Auto-generated method stub

    }

    public void onProcessUnstructuredSSRequest(ProcessUnstructuredSSRequest
procUnstrReqInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSRequest(UnstructuredSSRequest unstrReqInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSResponse(UnstructuredSSResponse unstrResInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyRequest(UnstructuredSSNotifyRequest
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyResponse(UnstructuredSSNotifyResponse
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }
```

```
    public void onDialogDelimiter(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequest(MAPDialog mapDialog, AddressString destReference,
AddressString origReference,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequestEricsson(MAPDialog mapDialog, AddressString
destReference, AddressString origReference,IMSI eriImsi, AddressString eriVlrNo) {
        // TODO Auto-generated method stub

    }

    public void onDialogAccept(MAPDialog mapDialog, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogUserAbort(MAPDialog mapDialog, MAPUserAbortChoice userReason
,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogProviderAbort(MAPDialog mapDialog, MAPAbortProviderReason
abortProviderReason,MAPAbortSource abortSource, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogClose(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogNotice(MAPDialog mapDialog, MAPNoticeProblemDiagnostic
noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    public void onDialogRelease(MAPDialog mapDialog) {
    }

    public void onDialogTimeout(MAPDialog mapDialog) {
```

```
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogReject(MAPDialog mapDialog, MAPRefuseReason refuseReason
,ApplicationContextName alternativeApplicationContext, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }
}
```

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

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.mobicents.protocols.ss7.map.api.MAPDialog;
import org.mobicents.protocols.ss7.map.api.MAPDialogListener;
import org.mobicents.protocols.ss7.map.api.MAPException;
import org.mobicents.protocols.ss7.map.api.MAPMessage;
import org.mobicents.protocols.ss7.map.api.MAPParameterFactory;
import org.mobicents.protocols.ss7.map.api.MAPProvider;
import org.mobicents.protocols.ss7.map.api.datacoding.CBSDataCodingScheme;
import org.mobicents.protocols.ss7.map.api.dialog.MAPAbortProviderReason;
import org.mobicents.protocols.ss7.map.api.dialog.MAPAbortSource;
import org.mobicents.protocols.ss7.map.api.dialog.MAPNoticeProblemDiagnostic;
import org.mobicents.protocols.ss7.map.api.dialog.MAPRefuseReason;
import org.mobicents.protocols.ss7.map.api.dialog.MAPUserAbortChoice;
import org.mobicents.protocols.ss7.map.api.errors.MAPErrorMessage;
import org.mobicents.protocols.ss7.map.api.primitives.AddressString;
import org.mobicents.protocols.ss7.map.api.primitives.IMSI;
import org.mobicents.protocols.ss7.map.api.primitives.MAPExtensionContainer;
import org.mobicents.protocols.ss7.map.api.primitives.USSDString;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.MAPDialogSupplementary;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListe
ner;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRequest
;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRespons
e;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyRequest;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyResponse
```

```
;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSRequest;
import
org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSResponse;
import org.mobicents.protocols.ss7.map.datacoding.CBSDataCodingSchemeImpl;
import org.mobicents.protocols.ss7.tcap.asn.ApplicationContextName;
import org.mobicents.protocols.ss7.tcap.asn.comp.Problem;

/**
 * A simple example show-casing how to use MAP stack. Demonstrates how to listen
 * to incoming Dialog from peer and process the MAP messages and send response.
 *
 * @author Amit Bhayani
 *
 */
public class UssdServerExample implements MAPDialogListener,
MAPServiceSupplementaryListener {

    private MAPProvider mapProvider;
    private MAPParameterFactory paramFact;

    public UssdServerExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {String providerJndiName = "java:/restcomm/ss7/map";this.mapProvider =
((MAPProvider) ctx.lookup(providerJndiName));
        } finally {ctx.close();
        }
    }

    public MAPProvider getMAPProvider() {
        return mapProvider;
    }

    public void start() {
        // Listen for Dialog events
        mapProvider.addMAPDialogListener(this);
        // Listen for USSD related messages
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);

        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
    }

    public void stop() {
        mapProvider.getMAPServiceSupplementary().deactivate();
    }

    public void onProcessUnstructuredSSRequest(ProcessUnstructuredSSRequest ind) {

        USSDString ussdString = ind.getUSSDString();
```

```
        MAPDialogSupplementary currentMapDialog = ind.getMAPDialog();

        try {String request = ussdString.getString(null);
// processing USSD requestString response = "Your balans is 100$";
CBSDataCodingScheme ussdDataCodingScheme = new CBSDataCodingSchemeImpl(0x0f
);USSDString ussdResponse = paramFact.createUSSDString(response, null, null);
currentMapDialog.addProcessUnstructuredSSResponse(ind.getInvokeId(),
ussdDataCodingScheme, ussdResponse);
        } catch (MAPException e1) {// TODO Auto-generated catch
blocke1.printStackTrace();
        }
    }

    public void onDialogDelimiter(MAPDialog mapDialog) {
        // This will initiate the TC-END with ReturnResultLast component
        try {mapDialog.send();
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
    }

    public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage) {
        // TODO Auto-generated method stub

    }

    public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem,
boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    public void onInvokeTimeout(MAPDialog mapDialog, Long invokeId) {
        // TODO Auto-generated method stub

    }

    public void onMAPMessage(MAPMessage mapMessage) {
        // TODO Auto-generated method stub

    }

    public void onProcessUnstructuredSSResponse(ProcessUnstructuredSSResponse ind) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSRequest(UnstructuredSSRequest unstrReqInd) {
        // TODO Auto-generated method stub
```

```
    }

    public void onUnstructuredSSResponse(UnstructuredSSResponse unstrResInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyRequest(UnstructuredSSNotifyRequest
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyResponse(UnstructuredSSNotifyResponse
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequest(MAPDialog mapDialog, AddressString destReference,
AddressString origReference,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequestEricsson(MAPDialog mapDialog, AddressString
destReference, AddressString origReference,IMSI eriImsi, AddressString eriVlrNo) {
        // TODO Auto-generated method stub

    }

    public void onDialogAccept(MAPDialog mapDialog, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogReject(MAPDialog mapDialog, MAPRefuseReason refuseReason
,ApplicationContextName alternativeApplicationContext, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogUserAbort(MAPDialog mapDialog, MAPUserAbortChoice userReason
,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogProviderAbort(MAPDialog mapDialog, MAPAbortProviderReason
```

```
abortProviderReason,MAPAbortSource abortSource, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogClose(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogNotice(MAPDialog mapDialog, MAPNoticeProblemDiagnostic
noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    public void onDialogRelease(MAPDialog mapDialog) {
    }

    public void onDialogTimeout(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }
}
```

Chapter 13. CAP

The (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. 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 this chapter you must be familiar with the specifications defined in 3GPP TS 22.078 (service aspects) and 3GPP TS 23.078 (technical realization).

Restcomm jSS7 CAP has full implementation for Phase 1 and Phase 2. The list of implemented operations can be found at CAPMessagesImplemented.ods. You can find the list of implemented operations here: CAPMessagesImplemented.ods. The document uses color coding to represent the status of the implementation:

Green

Fully Implemented

Red

Interface available (but not yet implemented)

Yellow

Implementation in progress

Blue

Implemeted only for CAP V2 and not available for CAP V3 and V4

Restcomm jSS7 Any contribution to implement other messages are welcome. We will provide you with all the help that you may require initially.

13.1. jSS7 CAP

The org.mobicents.protocols.ss7.cap.api.CAPStackinterface defines the methods required to represent CAP Protocol Stack. CAPStack exposes org.mobicents.protocols.ss7.cap.api.CAPProvider that interacts directly with CAPStack. This interface defines the methods that will be used by any registered CAP-User application. CAP-User application must implement interfaces to listen for CAP messages and for dialogue and component handling primitives. One interface is org.mobicents.protocols.ss7.cap.api.CAPDialogListener (listening associated with dialog events). Others are one or more interfaces for listening to messages associated with component events based on org.mobicents.protocols.ss7.cap.api.CAPServiceListener interface.

Each CAP-User interested in listening to messages specific to a CAP Service must implement the

specific `CAPServiceListener`.

- CAP-User interested only in circuit switched call operations implements `org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPServiceCircuitSwitchedCallListener`
- CAP-User interested only in GPRS operations implements `org.mobicents.protocols.ss7.cap.api.service.gprs.CAPServiceGprsListener`
- CAP-User interested only in SMS operations implements `org.mobicents.protocols.ss7.cap.api.service.sms.CAPServiceSmsListener`

CAP-User interested in all the services must implement all the service listener classes.

The `org.mobicents.protocols.ss7.cap.CAPStackImpl` is a concrete implementation of `CAPStack`. The CAP-User application gets access to `CAPProvider` by doing a JNDI look-up as explained in SS7 Service.

```
String providerJndiName = "java:/restcomm/ss7/cap";
this.capProvider = ((CAPProvider) ctx.lookup(providerJndiName));
...
```

The CAP-User application should register the concrete implementation of `CAPDialogListener` with `CAPProvider` to listen for incoming CAP Dialog and CAP Primitive messages. The CAP-User application should register the concrete implementation of `CAPServiceListener` with corresponding `CAPServiceBase ` to listen for incoming CAP Service messages. Following CAPServiceBase are exposed by [class]`CAPProvider`.

- For circuit switched call service `org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPServiceCircuitSwitchedCall`
- For GPRS service `org.mobicents.protocols.ss7.cap.api.service.gprs.CAPServiceGprs`
- For SMS service `org.mobicents.protocols.ss7.cap.api.service.sms.CAPServiceSms`

```
public class CAPExample implements CAPDialogListener,
CAPServiceCircuitSwitchedCallListener {
        ....
        capProvider.addCAPDialogListener(this);
        capProvider.getCAPServiceCircuitSwitchedCall().addCAPServiceListener(this);
        ....
}
```

Before any CAP specific service can be used, the corresponding service should be activated as shown below:

```
....
// Make the circuitSwitchedCall service activated
capProvider.getCAPServiceCircuitSwitchedCall().acivate();
....
```

The CAP-User Application leverages:

- `MAPParameterFactory` to create instances of `IMSI`, `ISDNAddressString`and many other MAP primitives that are used by CAP services.
- `CAPParameterFactory` to create instances of CAP primitives that are used by CAP services.
- `ISUPParameterFactory` to create instances of ISUP primitives that are used by CAP services.
- `INAPPParameterFactory` to create instances of INAP primitives that are used by CAP services.
- `CAPErrorMessageFactory` to create instances of CAP error messages like `CAPErrorMessageSystemFailure`.

```
MapParameterFactory mapParamFact = capProvider.getMapServiceFactory();
CapParameterFactory capParamFact = capProvider.getCapServiceFactory();
ISDNAddressString msisdn = mapParamFact.createISDNAddressString(
        AddressNature.international_number, NumberingPlan.ISDN,
        "31628838002");
BCSMEvent ev = capParamFact.createBCSMEvent(EventTypeBCSM.routeSelectFailure,
        MonitorMode.notifyAndContinue, null, null, false);
```

13.2. jSS7 Sending a CAP request (InitialDPRequest as an example)

For sending a CAP request you must do the following at the SSF side:

- Create a new CAP Dialog

```
// First create Dialog
SccpAddress origAddress = createLocalAddress();
SccpAddress destAddress = createRemoteAddress();
CAPApplicationContext acn = CAPApplicationContext.CapV2_gsmSSF_to_gsmSCF;
currentCapDialog = capProvider.getCAPServiceCircuitSwitchedCall().
    createNewDialog(acn, origAddress, remoteAddress);
```

- Add an Invoke component (InitialDPRequest message)

```
int serviceKey = 1;

CalledPartyNumber calledPartyNumber = client.getCAPProvider().
    getISUPParameterFactory().createCalledPartyNumber();
calledPartyNumber.setAddress("552348762");
calledPartyNumber.setNatureOfAddresIndicator(NAINumber._NAI_INTERNATIONAL_NUMBER);
calledPartyNumber.setNumberingPlanIndicator(CalledPartyNumber._NPI_ISDN);
calledPartyNumber.setInternalNetworkNumberIndicator(CalledPartyNumber._INN_ROUTING_ALL
OWED);
CalledPartyNumberCap calledPartyNumberCap = client.getCAPProvider().
    getCAPParameterFactory().createCalledPartyNumberCap(calledPartyNumber);

CallingPartyNumber callingPartyNumber = client.getCAPProvider().
    getISUPParameterFactory().createCallingPartyNumber();
callingPartyNumber.setAddress("55998223");
callingPartyNumber.setNatureOfAddresIndicator(NAINumber._NAI_INTERNATIONAL_NUMBER);
callingPartyNumber.setNumberingPlanIndicator(CalledPartyNumber._NPI_ISDN);
callingPartyNumber.setAddressRepresentationREstrictedIndicator(CallingPartyNumber._APR
I_ALLOWED);
callingPartyNumber.setScreeningIndicator(CallingPartyNumber._SI_NETWORK_PROVIDED);
CallingPartyNumberCap callingPartyNumberCap = client.getCAPProvider().
    getCAPParameterFactory().createCallingPartyNumberCap(callingPartyNumber);

LocationNumber locationNumber = client.getCAPProvider().getISUPParameterFactory().
    createLocationNumber();
locationNumber.setAddress("55200001");
locationNumber.setNatureOfAddresIndicator(NAINumber._NAI_INTERNATIONAL_NUMBER);
locationNumber.setNumberingPlanIndicator(LocationNumber._NPI_ISDN);
locationNumber.setAddressRepresentationRestrictedIndicator(LocationNumber._APRI_ALLOWE
D);
locationNumber.setScreeningIndicator(LocationNumber._SI_NETWORK_PROVIDED);
locationNumber.setInternalNetworkNumberIndicator(LocationNumber._INN_ROUTING_ALLOWED);
LocationNumberCap locationNumberCap = client.getCAPProvider().
getCAPParameterFactory().
    createLocationNumberCap(locationNumber);

ISDNAddressString vlrNumber = client.getCAPProvider().getMAPParameterFactory()
        .createISDNAddressString(AddressNature.international_number,
        NumberingPlan.ISDN, "552000002");
LocationInformation locationInformation = client.getCAPProvider
().getMAPParameterFactory()
        .createLocationInformation(10, null, vlrNumber, null,
        null, null, null, vlrNumber, null, false, false, null, null);

currentCapDialog.addInitialDPRequest(serviceKey, calledPartyNumber,
callingPartyNumber,
        null, null, null, locationNumber, null, null, null, null, null,
        eventTypeBCSM, null, null, null, null, null, null, null, false,
        null, null, locationInformation, null, null, null, null, null, false, null);
```

- Send a TC-Begin message to the server peer

```
// This will initiate the TC-BEGIN with INVOKE component
currentCapDialog.send();
```

- Wait for a response and other instructions from the SCF

At the SCF side when the TC-Begin message is received, the following sequence of events occur:

```
void CAPDialogListener.onDialogRequest(CAPDialog capDialog,
    CAPGprsReferenceNumber capGprsReferenceNumber);
```

This is the request for CAP Dialog processing. A CAP-User can reject the Dialog by invoking `capDialog.abort(CAPUserAbortReason abortReason)` method.

Then the incoming primitives corresponding events (one or more) occur. In this case it is

```
void CAPServiceCircuitSwitchedCallListener.onInitialDPRequest(InitialDPRequest ind);
```

When processing component-dependant messages you can add response components or add other Invoke requests. In this case it is RequestReportBCSMEventRequest as an example:

```
ArrayList<BCSMEvent> bcsmEventList = new ArrayList<BCSMEvent>();
BCSMEvent ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
    EventTypeBCSM.routeSelectFailure, MonitorMode.notifyAndContinue,
    null, null, false);
bcsmEventList.add(ev);
ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(EventTypeBCSM
.oCalledPartyBusy,
    MonitorMode.interrupted, null, null, false);
bcsmEventList.add(ev);
ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(EventTypeBCSM.
oNoAnswer,
    MonitorMode.interrupted, null, null, false);
bcsmEventList.add(ev);
ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(EventTypeBCSM.oAnswer,
    MonitorMode.notifyAndContinue, null, null, false);
bcsmEventList.add(ev);
LegID legId = this.capProvider.getINAPParameterFactory().createLegID(true, LegType
.leg1);
ev = this.capProvider.getCAPParameterFactory()
        .createBCSMEvent(EventTypeBCSM.oDisconnect, MonitorMode.notifyAndContinue,
legId, null, false);
bcsmEventList.add(ev);
legId = this.capProvider.getINAPParameterFactory().createLegID(true, LegType.leg2);
ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(EventTypeBCSM
.oDisconnect,
    MonitorMode.interrupted, legId, null, false);
bcsmEventList.add(ev);
ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(EventTypeBCSM.oAbandon,
    MonitorMode.notifyAndContinue, null, null, false);
bcsmEventList.add(ev);
currentCapDialog.addRequestReportBCSMEventRequest(bcsmEventList, null);

currentCapDialog.send();
```

If preparing the response takes time, you should return the control and prepare the answer in a separate thread.

If error or reject primitives are included into a TCAP message the following events occur:

```
public void onErrorComponent(CAPDialog capDialog, Long invokeId, CAPErrorMessage
capErrorMessage);
public void onProviderErrorComponent(CAPDialog capDialog, Long invokeId,
    CAPComponentErrorReason providerError);
public void onRejectComponent(CAPDialog capDialog, Long invokeId, Problem problem);
```

After all incoming components have been processed, the event `onDialogDelimiter(CAPDialog capDialog);` event is invoked (or `onDialogClose(CAPDialog capDialog)` in TC-END case). If all response components have been prepared you can tell the Stack to send response:

- `capDialog.close(false);` - to send TC-END
- `capDialog.send();` - to send TC-CONTINUE
- `capDialog.close(true);` - sends TC-END without any components (prearrangedEnd)

Instead of the methods `send()` and `close(boolean prearrangedEnd)`, you can invoke `sendDelayed()` or `closeDelayed(boolean prearrangedEnd)`. These methods are the same as `send()` and `close(boolean prearrangedEnd)` methods, but when invoking them from events of parsing incoming components real sending and dialog closing will occur only when all incoming component events and onDialogDelimiter() or onDialogClose() would be processed. If all response components have been prepared you should return the control and send a response when all components are ready.

In case of an error, you can terminate a CAP dialog in any method by invoking:

- `capDialog.abort(CAPUserAbortReason abortReason);` - sends TC-U-ABORT primitive

If there are no local actions or no response from a remote size for a long time, timeouts occur and the following methods are invoked:

- `CAPDialogListener.onDialogTimeout(CAPDialog capDialog);`
- `CAPServiceListener.onInvokeTimeout(CAPDialog capDialog, Long invokeId);`

In the `onDialogTimeout()` method you can invoke `capDialog.keepAlive();` to prevent a Dialog from closing. For preventing an Invoke timeout you should invoke `resetInvokeTimer(Long invokeId);` (before `onInvokeTimeout()` occurs).

13.3. jSS7 CAP Usage

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

import org.mobicents.protocols.ss7.cap.api.EsiBcsm.OAnswerSpecificInfo;
import org.mobicents.protocols.ss7.cap.api.EsiBcsm.ODisconnectSpecificInfo;
import org.mobicents.protocols.ss7.cap.api.isup.CalledPartyNumberCap;
import org.mobicents.protocols.ss7.cap.api.isup.CallingPartyNumberCap;
import org.mobicents.protocols.ss7.cap.api.isup.CauseCap;
import org.mobicents.protocols.ss7.cap.api.isup.LocationNumberCap;
import org.mobicents.protocols.ss7.cap.api.primitives.EventTypeBCSM;
import org.mobicents.protocols.ss7.cap.api.primitives.ReceivingSideID;
import org.mobicents.protocols.ss7.inap.api.primitives.LegType;
import org.mobicents.protocols.ss7.inap.api.primitives.MiscCallInfo;
import org.mobicents.protocols.ss7.inap.api.primitives.MiscCallInfoMessageType;
import org.mobicents.protocols.ss7.indicator.RoutingIndicator;
import org.mobicents.protocols.ss7.isup.message.parameter.CalledPartyNumber;
import org.mobicents.protocols.ss7.isup.message.parameter.CallingPartyNumber;
import org.mobicents.protocols.ss7.isup.message.parameter.CauseIndicators;
import org.mobicents.protocols.ss7.isup.message.parameter.LocationNumber;
import org.mobicents.protocols.ss7.isup.message.parameter.NAINumber;
import org.mobicents.protocols.ss7.map.api.primitives.AddressNature;
import org.mobicents.protocols.ss7.map.api.primitives.ISDNAddressString;
import org.mobicents.protocols.ss7.map.api.primitives.NumberingPlan;
```

```
import
org.mobicents.protocols.ss7.map.api.service.mobility.subscriberInformation.LocationInf
ormation;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;

public class Example {

    private static SccpAddress createLocalAddress() {
        return new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 1, null,
146);
    }

    private static SccpAddress createRemoteAddress() {
        return new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 2, null,
146);
    }


    public static void startCallSsf() throws Exception {
        CallSsfExample client = new CallSsfExample();

        client.start();

        // starting a call
        SccpAddress origAddress = createLocalAddress();
        SccpAddress destAddress = createRemoteAddress();

        int serviceKey = 1;

        CalledPartyNumber calledPartyNumber = client.getCAPProvider
().getISUPParameterFactory()
            .createCalledPartyNumber();
        calledPartyNumber.setAddress("552348762");
        calledPartyNumber.setNatureOfAddresIndicator(NAINumber
._NAI_INTERNATIONAL_NUMBER);
        calledPartyNumber.setNumberingPlanIndicator(CalledPartyNumber._NPI_ISDN);
        calledPartyNumber.setInternalNetworkNumberIndicator(CalledPartyNumber
._INN_ROUTING_ALLOWED);
        CalledPartyNumberCap calledPartyNumberCap = client.getCAPProvider()
            .getCAPParameterFactory().createCalledPartyNumberCap(calledPartyNumber);

        CallingPartyNumber callingPartyNumber = client.getCAPProvider
().getISUPParameterFactory()
            .createCallingPartyNumber();
        callingPartyNumber.setAddress("55998223");
        callingPartyNumber.setNatureOfAddresIndicator(NAINumber
._NAI_INTERNATIONAL_NUMBER);
        callingPartyNumber.setNumberingPlanIndicator(CalledPartyNumber._NPI_ISDN);
        callingPartyNumber.setAddressRepresentationREstrictedIndicator
(CallingPartyNumber._APRI_ALLOWED);
        callingPartyNumber.setScreeningIndicator(CallingPartyNumber
```

```
._SI_NETWORK_PROVIDED);
        CallingPartyNumberCap callingPartyNumberCap = client.getCAPProvider()
            .getCAPParameterFactory().createCallingPartyNumberCap(callingPartyNumber);

        LocationNumber locationNumber = client.getCAPProvider
().getISUPParameterFactory()
        .createLocationNumber();
        locationNumber.setAddress("55200001");
        locationNumber.setNatureOfAddresIndicator(NAINumber.
_NAI_INTERNATIONAL_NUMBER);
        locationNumber.setNumberingPlanIndicator(LocationNumber._NPI_ISDN);
        locationNumber.setAddressRepresentationRestrictedIndicator(LocationNumber
._APRI_ALLOWED);
        locationNumber.setScreeningIndicator(LocationNumber._SI_NETWORK_PROVIDED);
        locationNumber.setInternalNetworkNumberIndicator(LocationNumber
._INN_ROUTING_ALLOWED);
        LocationNumberCap locationNumberCap = client.getCAPProvider
().getCAPParameterFactory()
            .createLocationNumberCap(locationNumber);

        ISDNAddressString vlrNumber = client.getCAPProvider().getMAPParameterFactory()
                .createISDNAddressString(AddressNature.international_number,
NumberingPlan.ISDN,
                        "552000002");
        LocationInformation locationInformation = client.getCAPProvider
().getMAPParameterFactory()
                .createLocationInformation(10, null, vlrNumber, null, null, null,
null,
                        vlrNumber, null, false, false, null, null);

        client.sendInitialDP(origAddress, destAddress, serviceKey,
calledPartyNumberCap,
                callingPartyNumberCap, locationNumberCap, EventTypeBCSM.collectedInfo,
                locationInformation);

        // sending oAnswer in 5 sec
        Thread.sleep(5000);
        OAnswerSpecificInfo oAnswerSpecificInfo = client.getCAPProvider
().getCAPParameterFactory()
                .createOAnswerSpecificInfo(null, false, false, null, null, null);
        ReceivingSideID legID = client.getCAPProvider().getCAPParameterFactory()
            .createReceivingSideID(LegType.leg2);
        MiscCallInfo miscCallInfo = client.getCAPProvider().getINAPParameterFactory()
            .createMiscCallInfo(MiscCallInfoMessageType.notification, null);
        client.sendEventReportBCSM_OAnswer(oAnswerSpecificInfo, legID, miscCallInfo);

        // sending oDisconnect in 20 sec
        Thread.sleep(20000);
        CauseIndicators causeIndicators = client.getCAPProvider
().getISUPParameterFactory()
            .createCauseIndicators();
```

```
        causeIndicators.setLocation(CauseIndicators._LOCATION_USER);
        causeIndicators.setCodingStandard(CauseIndicators._CODING_STANDARD_ITUT);
        causeIndicators.setCauseValue(CauseIndicators._CV_ALL_CLEAR);
        CauseCap releaseCause = client.getCAPProvider().getCAPParameterFactory()
            .createCauseCap(causeIndicators);
        ODisconnectSpecificInfo oDisconnectSpecificInfo = client.getCAPProvider()
            .getCAPParameterFactory().createODisconnectSpecificInfo(releaseCause);
        legID = client.getCAPProvider().getCAPParameterFactory().
createReceivingSideID(LegType.leg1);
        miscCallInfo = client.getCAPProvider().getINAPParameterFactory()
            .createMiscCallInfo(MiscCallInfoMessageType.notification, null);
        client.sendEventReportBCSM_ODisconnect(oDisconnectSpecificInfo, legID,
miscCallInfo);

        // wait for answer
        Thread.sleep(600000);

        client.stop();
    }

    public static void startCallScf() throws Exception {
        CallScfExample server = new CallScfExample();

        server.start();

        // wait for a request
        Thread.sleep(600000);

        server.stop();
    }
}
```

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

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.mobicents.protocols.ss7.cap.api.CAPApplicationContext;
import org.mobicents.protocols.ss7.cap.api.CAPDialog;
import org.mobicents.protocols.ss7.cap.api.CAPDialogListener;
import org.mobicents.protocols.ss7.cap.api.CAPException;
import org.mobicents.protocols.ss7.cap.api.CAPMessage;
import org.mobicents.protocols.ss7.cap.api.CAPParameterFactory;
import org.mobicents.protocols.ss7.cap.api.CAPProvider;
import org.mobicents.protocols.ss7.cap.api.CAPStack;
import org.mobicents.protocols.ss7.cap.api.EsiBcsm.OAnswerSpecificInfo;
import org.mobicents.protocols.ss7.cap.api.EsiBcsm.ODisconnectSpecificInfo;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPGeneralAbortReason;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPGprsReferenceNumber;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPNoticeProblemDiagnostic;
```

```
import org.mobicents.protocols.ss7.cap.api.dialog.CAPUserAbortReason;
import org.mobicents.protocols.ss7.cap.api.errors.CAPErrorMessage;
import org.mobicents.protocols.ss7.cap.api.isup.CalledPartyNumberCap;
import org.mobicents.protocols.ss7.cap.api.isup.CallingPartyNumberCap;
import org.mobicents.protocols.ss7.cap.api.isup.LocationNumberCap;
import org.mobicents.protocols.ss7.cap.api.primitives.EventTypeBCSM;
import org.mobicents.protocols.ss7.cap.api.primitives.ReceivingSideID;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ActivityTestRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ActivityTestResponse;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ApplyChargingReportReq
uest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ApplyChargingRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.AssistRequestInstructi
onsRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPDialogCircuitSwitch
edCall;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPServiceCircuitSwitc
hedCallListener;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CallInformationReportR
equest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CallInformationRequest
Request;
import org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CancelRequest;
import org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ConnectRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ConnectToResourceReque
st;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ContinueRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.DisconnectForwardConne
ctionRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.EstablishTemporaryConn
ectionRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.EventReportBCSMRequest
;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.FurnishChargingInforma
tionRequest;
import
```

```
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.InitialDPRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PlayAnnouncementReques
t;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PromptAndCollectUserIn
formationRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PromptAndCollectUserIn
formationResponse;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ReleaseCallRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.RequestReportBCSMEvent
Request;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ResetTimerRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.SendChargingInformatio
nRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.SpecializedResourceRep
ortRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.primitive.DestinationR
outingAddress;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.primitive.EventSpecifi
cInformationBCSM;
import org.mobicents.protocols.ss7.inap.api.primitives.MiscCallInfo;
import org.mobicents.protocols.ss7.map.api.MAPProvider;
import
org.mobicents.protocols.ss7.map.api.service.mobility.subscriberInformation.LocationInf
ormation;
import org.mobicents.protocols.ss7.sccp.SccpProvider;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;
import org.mobicents.protocols.ss7.tcap.asn.comp.PAbortCauseType;
import org.mobicents.protocols.ss7.tcap.asn.comp.Problem;

public class CallSsfExample implements CAPDialogListener,
CAPServiceCircuitSwitchedCallListener {

    private CAPProvider capProvider;
    private CAPParameterFactory paramFact;
    private CAPDialogCircuitSwitchedCall currentCapDialog;
    private CallContent cc;

    public CallSsfExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {
            String providerJndiName = "java:/restcomm/ss7/cap";
```

```
            this.capProvider = ((CAPProvider) ctx.lookup(providerJndiName));
        } finally {
            ctx.close();
        }

        paramFact = capProvider.getCAPParameterFactory();

        capProvider.addCAPDialogListener(this);
        capProvider.getCAPServiceCircuitSwitchedCall().addCAPServiceListener(this);
    }

    public CAPProvider getCAPProvider() {
        return capProvider;
    }

    public void start() {
        // Make the circuitSwitchedCall service activated
        capProvider.getCAPServiceCircuitSwitchedCall().acivate();

        currentCapDialog = null;
    }

    public void stop() {
         capProvider.getCAPServiceCircuitSwitchedCall().deactivate();
    }

    public void sendInitialDP(SccpAddress origAddress, SccpAddress remoteAddress,
            int serviceKey, CalledPartyNumberCap calledPartyNumber,
            CallingPartyNumberCap callingPartyNumber, LocationNumberCap
locationNumber,
            EventTypeBCSM eventTypeBCSM, LocationInformation locationInformation)
            throws CAPException {
        // First create Dialog
        CAPApplicationContext acn = CAPApplicationContext.CapV2_gsmSSF_to_gsmSCF;
        currentCapDialog = capProvider.getCAPServiceCircuitSwitchedCall
().createNewDialog(
                acn, origAddress, remoteAddress);

        currentCapDialog.addInitialDPRequest(serviceKey, calledPartyNumber,
                callingPartyNumber, null, null, null, locationNumber, null, null,
                null, null, null,
                eventTypeBCSM, null, null, null, null, null, null, null, false,
                null, null, locationInformation, null, null, null, null, null, false,
null);
        // This will initiate the TC-BEGIN with INVOKE component
        currentCapDialog.send();

        this.cc.step = Step.initialDPSent;
        this.cc.calledPartyNumber = calledPartyNumber;
        this.cc.callingPartyNumber = callingPartyNumber;
```

```
    }

    public void sendEventReportBCSM_OAnswer(OAnswerSpecificInfo oAnswerSpecificInfo,
            ReceivingSideID legID, MiscCallInfo miscCallInfo) throws CAPException {
        if (currentCapDialog != null && this.cc != null) {
            EventSpecificInformationBCSM eventSpecificInformationBCSM =
                this.capProvider.getCAPParameterFactory
().createEventSpecificInformationBCSM(
                        oAnswerSpecificInfo);
            currentCapDialog.addEventReportBCSMRequest(EventTypeBCSM.oAnswer,
                    eventSpecificInformationBCSM, legID, miscCallInfo, null);
            currentCapDialog.send();
            this.cc.step = Step.answered;
        }
    }

    public void sendEventReportBCSM_ODisconnect(ODisconnectSpecificInfo
oDisconnectSpecificInfo,
            ReceivingSideID legID, MiscCallInfo miscCallInfo) throws CAPException {
        if (currentCapDialog != null && this.cc != null) {
            EventSpecificInformationBCSM eventSpecificInformationBCSM =
                this.capProvider.getCAPParameterFactory
().createEventSpecificInformationBCSM(
                        oDisconnectSpecificInfo);
            currentCapDialog.addEventReportBCSMRequest(EventTypeBCSM.oDisconnect,
                    eventSpecificInformationBCSM, legID, miscCallInfo, null);
            currentCapDialog.send();
            this.cc.step = Step.disconnected;
        }
    }

    @Override
    public void onRequestReportBCSMEventRequest(RequestReportBCSMEventRequest ind) {
        if (currentCapDialog != null && this.cc != null && this.cc.step != Step
.disconnected) {
            this.cc.requestReportBCSMEventRequest = ind;

            // initiating BCSM events processing
        }
        ind.getCAPDialog().processInvokeWithoutAnswer(ind.getInvokeId());
    }

    @Override
    public void onActivityTestRequest(ActivityTestRequest ind) {
        if (currentCapDialog != null && this.cc != null && this.cc.step != Step
.disconnected) {
            this.cc.activityTestInvokeId = ind.getInvokeId();
        }
    }

    @Override
```

```
    public void onActivityTestResponse(ActivityTestResponse ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onContinueRequest(ContinueRequest ind) {
        this.cc.step = Step.callAllowed;
        ind.getCAPDialog().processInvokeWithoutAnswer(ind.getInvokeId());
        // sending Continue to use the original calledPartyAddress
    }

    @Override
    public void onConnectRequest(ConnectRequest ind) {
        this.cc.step = Step.callAllowed;
        this.cc.destinationRoutingAddress = ind.getDestinationRoutingAddress();
        ind.getCAPDialog().processInvokeWithoutAnswer(ind.getInvokeId());
        // sending Connect to force routing the call to a new  number
    }

    @Override
    public void onDialogTimeout(CAPDialog capDialog) {
        if (currentCapDialog != null && this.cc != null && this.cc.step != Step
.disconnected) {
            // if the call is still up - keep the sialog alive
            currentCapDialog.keepAlive();
        }
    }

    @Override
    public void onDialogDelimiter(CAPDialog capDialog) {
        if (currentCapDialog != null && this.cc != null && this.cc.step != Step
.disconnected) {
            if (this.cc.activityTestInvokeId != null) {
                try {
                    currentCapDialog.addActivityTestResponse(this.cc
.activityTestInvokeId);
                    this.cc.activityTestInvokeId = null;
                    currentCapDialog.send();
                } catch (CAPException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }

    @Override
    public void onErrorComponent(CAPDialog capDialog, Long invokeId, CAPErrorMessage
capErrorMessage) {
        // TODO Auto-generated method stub
```

```
    }

    @Override
    public void onRejectComponent(CAPDialog capDialog, Long invokeId, Problem problem,
            boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onInvokeTimeout(CAPDialog capDialog, Long invokeId) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCAPMessage(CAPMessage capMessage) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onInitialDPRequest(InitialDPRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onApplyChargingRequest(ApplyChargingRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onEventReportBCSMRequest(EventReportBCSMRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onApplyChargingReportRequest(ApplyChargingReportRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onReleaseCallRequest(ReleaseCallRequest ind) {
        // TODO Auto-generated method stub

    }
```

```
    @Override
    public void onCallInformationRequestRequest(CallInformationRequestRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCallInformationReportRequest(CallInformationReportRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onAssistRequestInstructionsRequest(AssistRequestInstructionsRequest
ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onEstablishTemporaryConnectionRequest
(EstablishTemporaryConnectionRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDisconnectForwardConnectionRequest
(DisconnectForwardConnectionRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onConnectToResourceRequest(ConnectToResourceRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onResetTimerRequest(ResetTimerRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onFurnishChargingInformationRequest(FurnishChargingInformationRequest
ind) {
        // TODO Auto-generated method stub
```

```
    }

    @Override
    public void onSendChargingInformationRequest(SendChargingInformationRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onSpecializedResourceReportRequest(SpecializedResourceReportRequest
ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPlayAnnouncementRequest(PlayAnnouncementRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPromptAndCollectUserInformationRequest
(PromptAndCollectUserInformationRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPromptAndCollectUserInformationResponse
(PromptAndCollectUserInformationResponse ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCancelRequest(CancelRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogRequest(CAPDialog capDialog, CAPGprsReferenceNumber
capGprsReferenceNumber) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogAccept(CAPDialog capDialog, CAPGprsReferenceNumber
capGprsReferenceNumber) {
```

```
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogUserAbort(CAPDialog capDialog, CAPGeneralAbortReason
generalReason,
            CAPUserAbortReason userReason) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogProviderAbort(CAPDialog capDialog, PAbortCauseType abortCause)
{
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogClose(CAPDialog capDialog) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogRelease(CAPDialog capDialog) {
        this.currentCapDialog = null;
        this.cc = null;
    }

    @Override
    public void onDialogNotice(CAPDialog capDialog, CAPNoticeProblemDiagnostic
noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    private enum Step {
        initialDPSent,
        callAllowed,
        answered,
        disconnected;
    }

    private class CallContent {
        public Step step;
        public Long activityTestInvokeId;

        public CalledPartyNumberCap calledPartyNumber;
        public CallingPartyNumberCap callingPartyNumber;
```

```
        public RequestReportBCSMEventRequest requestReportBCSMEventRequest;
        public DestinationRoutingAddress destinationRoutingAddress;
    }

}
```

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

import java.util.ArrayList;

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.mobicents.protocols.ss7.cap.api.CAPDialog;
import org.mobicents.protocols.ss7.cap.api.CAPDialogListener;
import org.mobicents.protocols.ss7.cap.api.CAPException;
import org.mobicents.protocols.ss7.cap.api.CAPMessage;
import org.mobicents.protocols.ss7.cap.api.CAPParameterFactory;
import org.mobicents.protocols.ss7.cap.api.CAPProvider;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPGeneralAbortReason;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPGprsReferenceNumber;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPNoticeProblemDiagnostic;
import org.mobicents.protocols.ss7.cap.api.dialog.CAPUserAbortReason;
import org.mobicents.protocols.ss7.cap.api.errors.CAPErrorMessage;
import org.mobicents.protocols.ss7.cap.api.isup.CalledPartyNumberCap;
import org.mobicents.protocols.ss7.cap.api.primitives.BCSMEvent;
import org.mobicents.protocols.ss7.cap.api.primitives.EventTypeBCSM;
import org.mobicents.protocols.ss7.cap.api.primitives.MonitorMode;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ActivityTestRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ActivityTestResponse;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ApplyChargingReportReq
uest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ApplyChargingRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.AssistRequestInstructi
onsRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPDialogCircuitSwitch
edCall;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CAPServiceCircuitSwitc
hedCallListener;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CallInformationReportR
equest;
import
```

```
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CallInformationRequest
Request;
import org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.CancelRequest;
import org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ConnectRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ConnectToResourceReque
st;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ContinueRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.DisconnectForwardConne
ctionRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.EstablishTemporaryConn
ectionRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.EventReportBCSMRequest
;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.FurnishChargingInforma
tionRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.InitialDPRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PlayAnnouncementReques
t;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PromptAndCollectUserIn
formationRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.PromptAndCollectUserIn
formationResponse;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ReleaseCallRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.RequestReportBCSMEvent
Request;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.ResetTimerRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.SendChargingInformatio
nRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.SpecializedResourceRep
ortRequest;
import
org.mobicents.protocols.ss7.cap.api.service.circuitSwitchedCall.primitive.DestinationR
outingAddress;
import org.mobicents.protocols.ss7.inap.api.primitives.LegID;
import org.mobicents.protocols.ss7.inap.api.primitives.LegType;
import org.mobicents.protocols.ss7.isup.message.parameter.CalledPartyNumber;
```

```
import org.mobicents.protocols.ss7.isup.message.parameter.NAINumber;
import org.mobicents.protocols.ss7.tcap.asn.comp.PAbortCauseType;
import org.mobicents.protocols.ss7.tcap.asn.comp.Problem;

public class CallScfExample implements CAPDialogListener,
CAPServiceCircuitSwitchedCallListener {

    private CAPProvider capProvider;
    private CAPParameterFactory paramFact;
    private CAPDialogCircuitSwitchedCall currentCapDialog;
    private CallContent cc;

    public CallScfExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {
            String providerJndiName = "java:/restcomm/ss7/cap";
            this.capProvider = ((CAPProvider) ctx.lookup(providerJndiName));
        } finally {
            ctx.close();
        }
        paramFact = capProvider.getCAPParameterFactory();

        capProvider.addCAPDialogListener(this);
        capProvider.getCAPServiceCircuitSwitchedCall().addCAPServiceListener(this);
    }

    public CAPProvider getCAPProvider() {
        return capProvider;
    }

    public void start() {
        // Make the circuitSwitchedCall service activated
        capProvider.getCAPServiceCircuitSwitchedCall().acivate();

        currentCapDialog = null;
    }

    public void stop() {
        capProvider.getCAPServiceCircuitSwitchedCall().deactivate();
    }

    @Override
    public void onInitialDPRequest(InitialDPRequest ind) {
        this.cc = new CallContent();
        this.cc.idp = ind;
        this.cc.step = Step.initialDPRecieved;

        ind.getCAPDialog().processInvokeWithoutAnswer(ind.getInvokeId());
    }

    @Override
```

```
    public void onEventReportBCSMRequest(EventReportBCSMRequest ind) {
        if (this.cc != null) {
            this.cc.eventList.add(ind);

            switch (ind.getEventTypeBCSM()) {
            case oAnswer:
                this.cc.step = Step.answered;
                break;
            case oDisconnect:
                this.cc.step = Step.disconnected;
                break;
            }
        }

        ind.getCAPDialog().processInvokeWithoutAnswer(ind.getInvokeId());
    }

    @Override
    public void onDialogDelimiter(CAPDialog capDialog) {
        try {
            if (this.cc != null) {
                switch (this.cc.step) {
                case initialDPRecieved:
                    // informing SSF of BCSM events processing
                    ArrayList<BCSMEvent> bcsmEventList = new ArrayList<BCSMEvent>();
                    BCSMEvent ev = this.capProvider.getCAPParameterFactory
().createBCSMEvent(
                            EventTypeBCSM.routeSelectFailure, MonitorMode
.notifyAndContinue,
                            null, null, false);
                    bcsmEventList.add(ev);
                    ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
                            EventTypeBCSM.oCalledPartyBusy, MonitorMode.interrupted,
null,
                            null, false);
                    bcsmEventList.add(ev);
                    ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
                            EventTypeBCSM.oNoAnswer, MonitorMode.interrupted, null,
null, false);
                    bcsmEventList.add(ev);
                    ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
                            EventTypeBCSM.oAnswer, MonitorMode.notifyAndContinue,
null,
                            null, false);
                    bcsmEventList.add(ev);
                    LegID legId = this.capProvider.getINAPParameterFactory
().createLegID(
                            true, LegType.leg1);
                    ev = this.capProvider.getCAPParameterFactory()
                            .createBCSMEvent(EventTypeBCSM.oDisconnect,
                                    MonitorMode.notifyAndContinue, legId, null,
```

```
false);
                    bcsmEventList.add(ev);
                    legId = this.capProvider.getINAPParameterFactory().createLegID
(true,
                            LegType.leg2);
                    ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
                            EventTypeBCSM.oDisconnect, MonitorMode.interrupted, legId,
                            null, false);
                    bcsmEventList.add(ev);
                    ev = this.capProvider.getCAPParameterFactory().createBCSMEvent(
                            EventTypeBCSM.oAbandon, MonitorMode.notifyAndContinue,
null,
                            null, false);
                    bcsmEventList.add(ev);
                    currentCapDialog.addRequestReportBCSMEventRequest(bcsmEventList,
null);

                    // calculating here a new called party number if it is needed
                    String newNumber = "22123124";
                    if (newNumber != null) {
                        // sending Connect to force routing the call to a new  number
                        ArrayList<CalledPartyNumberCap> calledPartyNumber =
                            new ArrayList<CalledPartyNumberCap>();
                        CalledPartyNumber cpn = this.capProvider
.getISUPParameterFactory()
                            .createCalledPartyNumber();
                        cpn.setAddress("5599999988");
                        cpn.setNatureOfAddresIndicator(NAINumber
._NAI_INTERNATIONAL_NUMBER);
                        cpn.setNumberingPlanIndicator(CalledPartyNumber._NPI_ISDN);
                        cpn.setInternalNetworkNumberIndicator(
                                CalledPartyNumber._INN_ROUTING_ALLOWED);
                        CalledPartyNumberCap cpnc = this.capProvider
.getCAPParameterFactory()
                        .createCalledPartyNumberCap(cpn);
                        calledPartyNumber.add(cpnc);
                        DestinationRoutingAddress destinationRoutingAddress = this
.capProvider
                            .getCAPParameterFactory().createDestinationRoutingAddress(
                                calledPartyNumber);
                        currentCapDialog.addConnectRequest(destinationRoutingAddress,
null,
                                null, null, null, null, null, null, null, null, null,
                                null, null,
                                false, false, false, null, false);
                    } else {
                        // sending Continue to use the original calledPartyAddress
                        currentCapDialog.addContinueRequest();
                    }

                    currentCapDialog.send();
```

```
                    break;

                case disconnected:
                    // the call is terminated - close dialog
                    currentCapDialog.close(false);
                    break;
                }
            }
        } catch (CAPException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    @Override
    public void onDialogTimeout(CAPDialog capDialog) {
        if (currentCapDialog != null && this.cc != null && this.cc.step != Step
.disconnected
                && this.cc.activityTestInvokeId == null) {
            // check the SSF if the call is still alive
            currentCapDialog.keepAlive();
            try {
                this.cc.activityTestInvokeId = currentCapDialog.
addActivityTestRequest();
                currentCapDialog.send();
            } catch (CAPException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    @Override
    public void onActivityTestResponse(ActivityTestResponse ind) {
        if (currentCapDialog != null && this.cc != null) {
            this.cc.activityTestInvokeId = null;
        }
    }

    @Override
    public void onInvokeTimeout(CAPDialog capDialog, Long invokeId) {
        if (currentCapDialog != null && this.cc != null) {
            if (this.cc.activityTestInvokeId == invokeId) { // activityTest failure
                try {
                    currentCapDialog.close(true);
                } catch (CAPException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
```

```
    }

    @Override
    public void onErrorComponent(CAPDialog capDialog, Long invokeId,
            CAPErrorMessage capErrorMessage) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onRejectComponent(CAPDialog capDialog, Long invokeId, Problem problem,
            boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCAPMessage(CAPMessage capMessage) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onRequestReportBCSMEventRequest(RequestReportBCSMEventRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onApplyChargingRequest(ApplyChargingRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onContinueRequest(ContinueRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onApplyChargingReportRequest(ApplyChargingReportRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onReleaseCallRequest(ReleaseCallRequest ind) {
        // TODO Auto-generated method stub

    }
```

```
    @Override
    public void onConnectRequest(ConnectRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCallInformationRequestRequest(CallInformationRequestRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCallInformationReportRequest(CallInformationReportRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onActivityTestRequest(ActivityTestRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onAssistRequestInstructionsRequest(AssistRequestInstructionsRequest
ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onEstablishTemporaryConnectionRequest
(EstablishTemporaryConnectionRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDisconnectForwardConnectionRequest
(DisconnectForwardConnectionRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onConnectToResourceRequest(ConnectToResourceRequest ind) {
        // TODO Auto-generated method stub

    }
```

```
    @Override
    public void onResetTimerRequest(ResetTimerRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onFurnishChargingInformationRequest(FurnishChargingInformationRequest
ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onSendChargingInformationRequest(SendChargingInformationRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onSpecializedResourceReportRequest(SpecializedResourceReportRequest
ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPlayAnnouncementRequest(PlayAnnouncementRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPromptAndCollectUserInformationRequest
(PromptAndCollectUserInformationRequest ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onPromptAndCollectUserInformationResponse
(PromptAndCollectUserInformationResponse ind) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onCancelRequest(CancelRequest ind) {
        // TODO Auto-generated method stub
```

```
    }

    @Override
    public void onDialogRequest(CAPDialog capDialog, CAPGprsReferenceNumber
capGprsReferenceNumber) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogAccept(CAPDialog capDialog, CAPGprsReferenceNumber
capGprsReferenceNumber) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogUserAbort(CAPDialog capDialog, CAPGeneralAbortReason
generalReason,
            CAPUserAbortReason userReason) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogProviderAbort(CAPDialog capDialog, PAbortCauseType abortCause)
{
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogClose(CAPDialog capDialog) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogRelease(CAPDialog capDialog) {
        this.currentCapDialog = null;
        this.cc = null;
    }

    @Override
    public void onDialogNotice(CAPDialog capDialog,
            CAPNoticeProblemDiagnostic noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    private enum Step {
```

```
        initialDPRecieved,
        answered,
        disconnected;
    }

    private class CallContent {
        public Step step;
        public InitialDPRequest idp;
        public ArrayList<EventReportBCSMRequest> eventList = new ArrayList
<EventReportBCSMRequest>();
        public Long activityTestInvokeId;
    }
}
```

Chapter 14. SS7 Simulator

SS7 Simulator is an application for testing the Restcomm jSS7 and understanding its functionality. The test cases will allow you to conduct various test scenarios and explore the Stack. The Simulator is also a good example of how to use this Stack. For details on installing the Simulator please refer to the Installation Guide. To know more about Running the Simulator please refer to Running Restcomm jSS7 Simulator in this guide.

Before running a test all relevant layers of the Simulator must be configured. After running a test, you can perform some actions depending on the test. The results of the test are emitted as "notifications". Notifications are displayed in the GUI interface of the Simulator and JConsole application (which is also a GUI interface). Notifications are also written into a file "<host name>.log" (for example "a1.log") as well as printed onto a console.

14.1. Configuring Restcomm jSS7 Simulator

You must configure all relevant layers and the Testing Task prior to running a test with the Simulator. For details on how to launch the form for configuring the simulator, refer to [_procedure_test_simulator].

14.1.1. SCCP Layer of the Simulator

SCCP layer supports message processing in two modes:

- Route on DPC and SSN mode
- Route on GlobalTitle mode

For both modes you must set the following options:

- Remote SPC
- Local SPC
- Network indicator
- Remote SSN
- Local SSN

When the SCCP layer is started, the following are created:

- Mtp3ServiceAccessPoint (using Local SPC and Network indicator) with a single Mtp3Destination (Remote SPC, all SLS's)
- RemoteSpc (using Remote SPC)
- RemoteSsn (using Remote SPC and Remote SSN)

For "Route on GlobalTitle" mode there are few extra options as below:

- GlobalTitle type ("Nature of address indicator only", "Translation type only", "Translation type, numbering plan and encoding scheme", "Translation type, numbering plan, encoding scheme

and NOA ind"), the last choice is default.

- AddressNature
- NumberingPlan
- Translation type (those four options are used for CalledPartyAddress, CallingPartyAddress and routing rules)
- CallingPartyAddress digits

If the "Route on GlobalTitle" mode is used, two rules for routing are created when the SCCP layer is started. All remotely originated messages are routed to a local user, All locally originated messages are routed to a remote user (to a peer with Remote SPC address).

SCCP layer provide CalledPartyAddress and CallingPartyAddress for upper layers and test cases. When "Route on DPC and SSN" mode is used, these are ROUTING_BASED_ON_DPC_AND_SSN with no GlobalTitle, pc=Local SPC (for CallingPartyAddress) or Remote SSN (for CalledPartyAddress), SSN=Local SSN and Remote SSN accordingly.

When "Route on GlobalTitle" mode ROUTING_BASED_ON_GT is used:

- CallingPartyAddress is created based on CallingPartyAddress digits and Local SSN.
- CalledPartyAddress is created using digits and SSN that are supplied by upper levels or test cases.

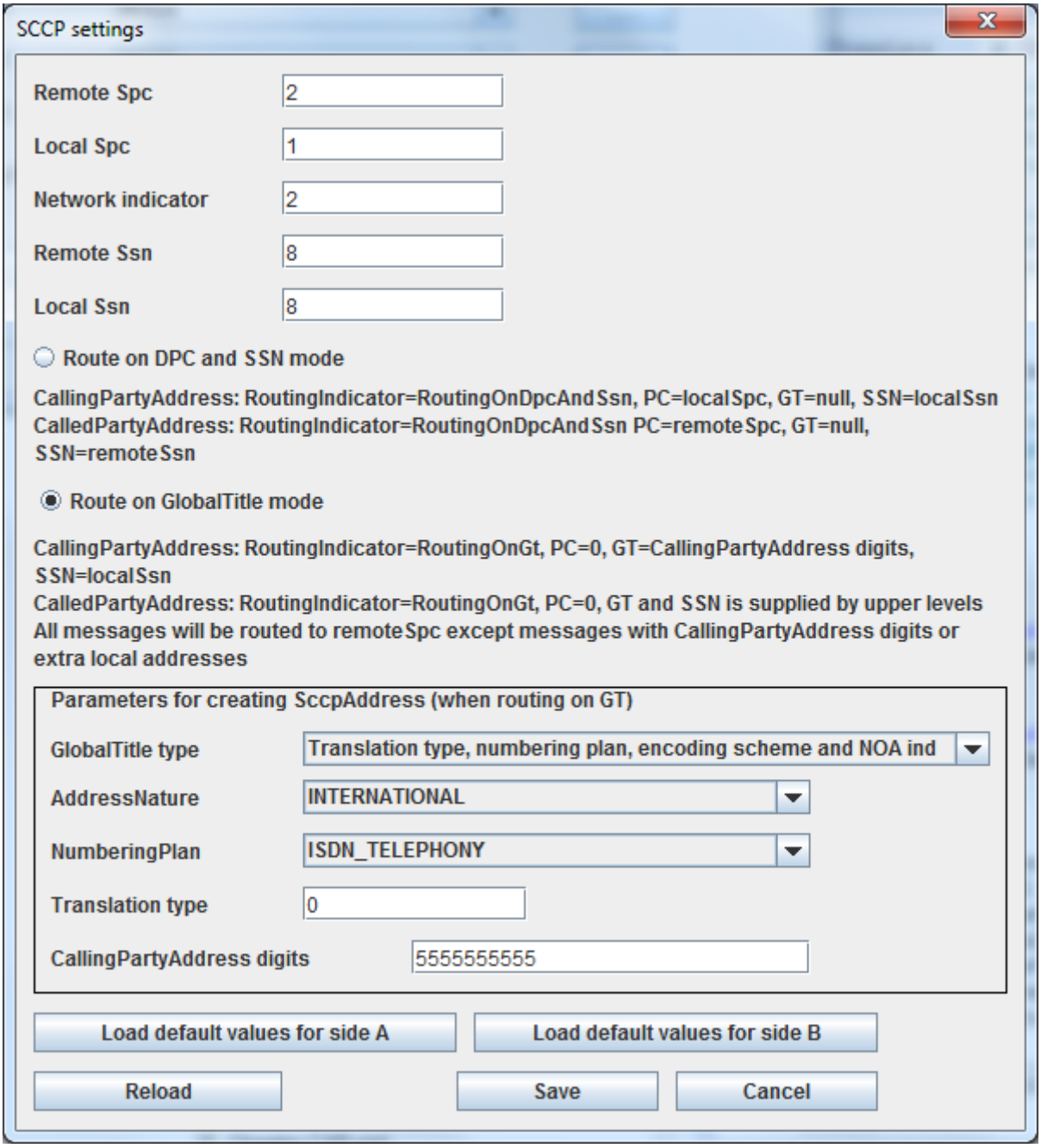


Figure 37. SCCP Settings Form

14.2. SS7 Simulator Test Cases

14.2.1. USSD Server

UssdServer test can work in manual and automatic modes. In the manual mode a user manually inserts the response message value and sends ProcessUnstructuredSs response (or UnstructuredSs request) to the Ussd client. In the automatic mode UssdServer automatically sends a predefined message as an ProcessUnstructuredSs response ("Auto sending ProcessUnstructuredSsResponse" option) or automatically sends a predefined message as an UnstructuredSs request and then sends a predefined message ProcessUnstructuredSs response ("Auto sending Unstructured_SS_Request then after response sending ProcessUnstructuredSsResponse" option). Use "Auto sending ProcessUnstructuredSsResponse" option for UssdServer if UssdClient is used for a load test ("Auto sending ProcessUnstructuredSSRequest"). For load tests we recommend to check the option "One

notification for 100 dialogs" for preventing too many notifications when load testing.

UssdServer test can perform followng tasks:

- Sending a ProcessUnstructuredSs response / UnstructuredSs request in automatic or manual modes. Ussd Client must initiate a dialog using ProcessUnstructuredSs request.
- Manually sending an UnstructuredSsNotify.

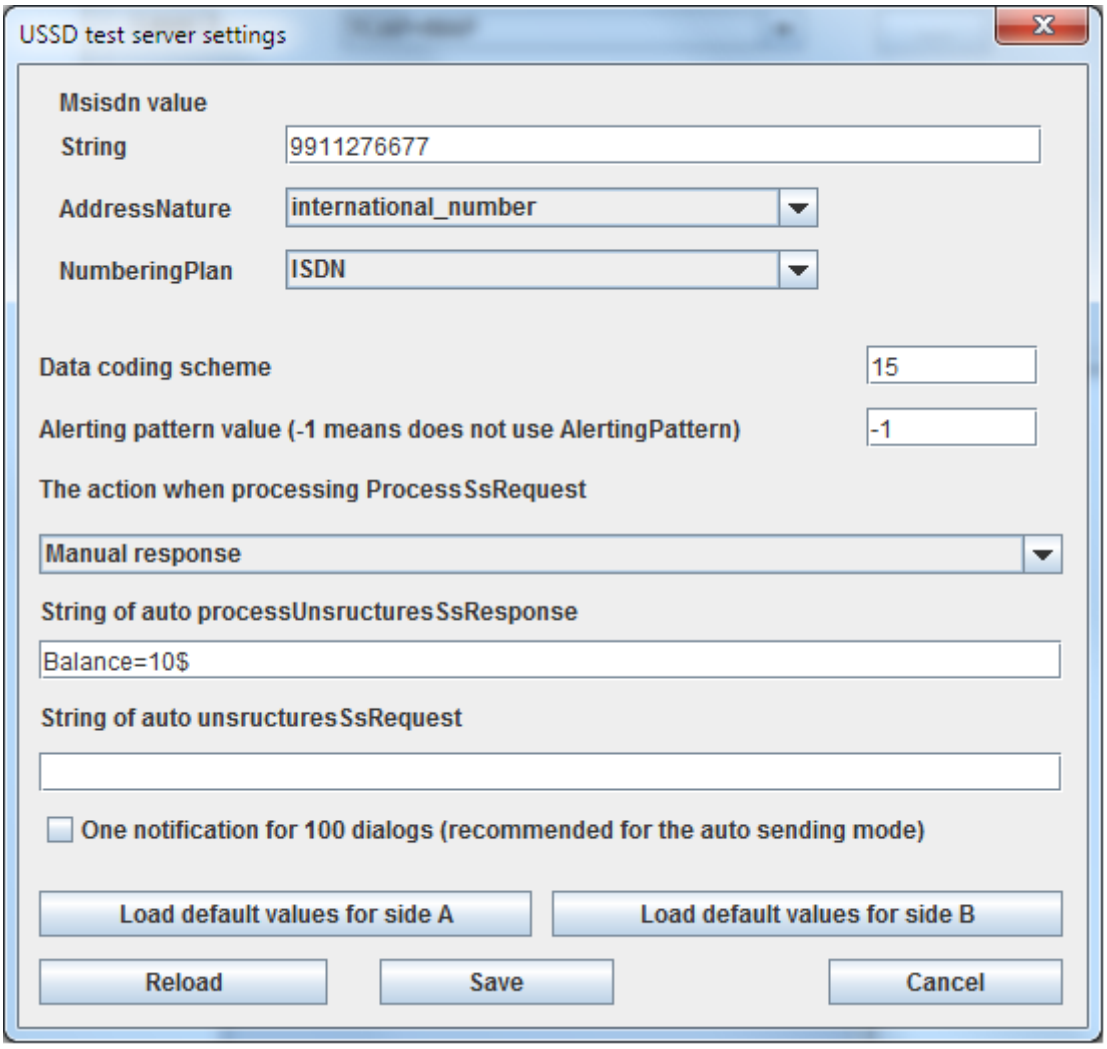


Figure 38. Ussd Server settings form

14.2.2. USSD Client

UssdClient test can perform following tasks:

- Sending a ProcessUnstructuredSs request, waiting for the answer and displaying the answer
- Sending a UnstructuredSs response as an answer for UnstructuredSs request
- Receiving UnstructuredSs notify and displaying it
- Special case for load testing: Sending to the Ussd server a flow of ProcessUnstructuredSs requests without stopping (and receiving responses).

For working in the manual mode select the option "Manual operation", for auto sending messages - the option "Auto sending ProcessUnstructuredSSRequest". You can send ProcessUnstructuredSs

request and UnstructuredSs response only manually by inserting a message text and pressing buttons. For working in the auto mode you should define the string of auto processUnstructuredSs request and the count of maximum active MAP dialogs (default value is 10). The more dialogs is defined the more messages per second will be sent. Msisdn, data coding schema and alerting pattern values should be also configured before test starting. For the auto mode we recomment to check the option "One notification for 100 dialogs" for preventing too many notifications when load testing.

You can send ProcessUnstructuredSs request and UnstructuredSs response manually by inserting a message text and pressing buttons. You can not send a new ProcessUnstructiredSs request till the response for previos request has been received (or till dialog timeout). You can also manually close the current dialog by pressing "Close current dialog" button.

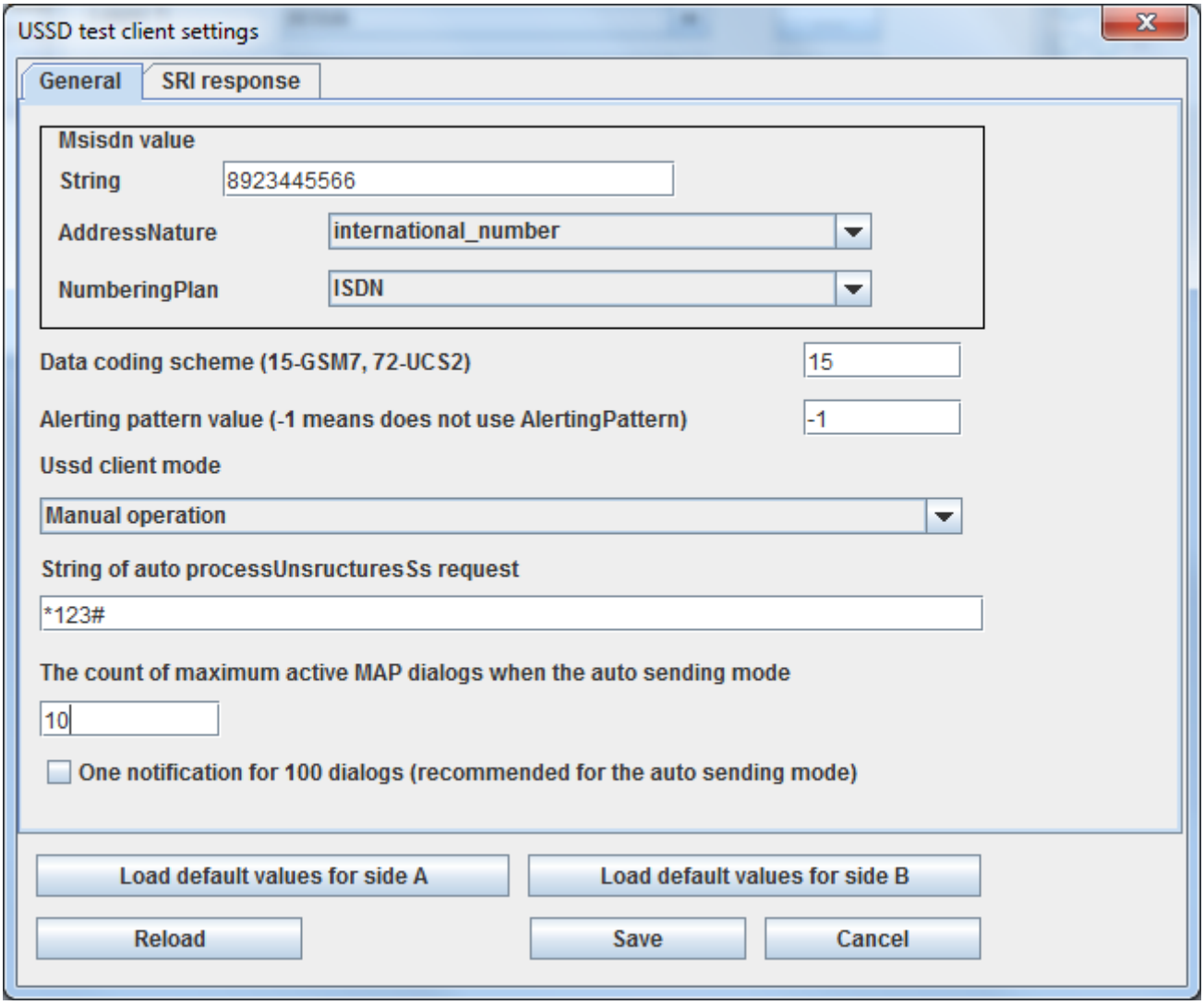


Figure 39. Ussd Client settings form

14.2.3. SMS Server

SMS Server simulates the behavior of SMSC, it can:

- Send sendRoutingInfoForSM to the HLR and receive the response
- Send mt-forwardSM to the VLR and receive the response
- Send sendRoutingInfoForSM to the HLR, receive the response and send mt-forwardSM using

data from the first request

- Receive mo-forwardSM request from VLR and display the received message

We need to set the following options to SMS server:

- AddressNature and NumberingPlan for AddressString creation
- TypeOfNumber and NumberingPlanIdentification SMS tpdu addresses
- MAP protocol version (1, 2 or 3) (version 3 is default)
- Character set for SMS message encoding (GSM7 or UCS2)
- Origination Service center address string (this address must be equal the SCCP layer CallingPartyAddress digits)
- HLR and VLR SSN values (default values are 8 and 6)

if you want to use SMS Server and SMS Client for sending SMS to each other: set HLR SSN at SMS Server the equal value that VLR SSN (8 in our case)

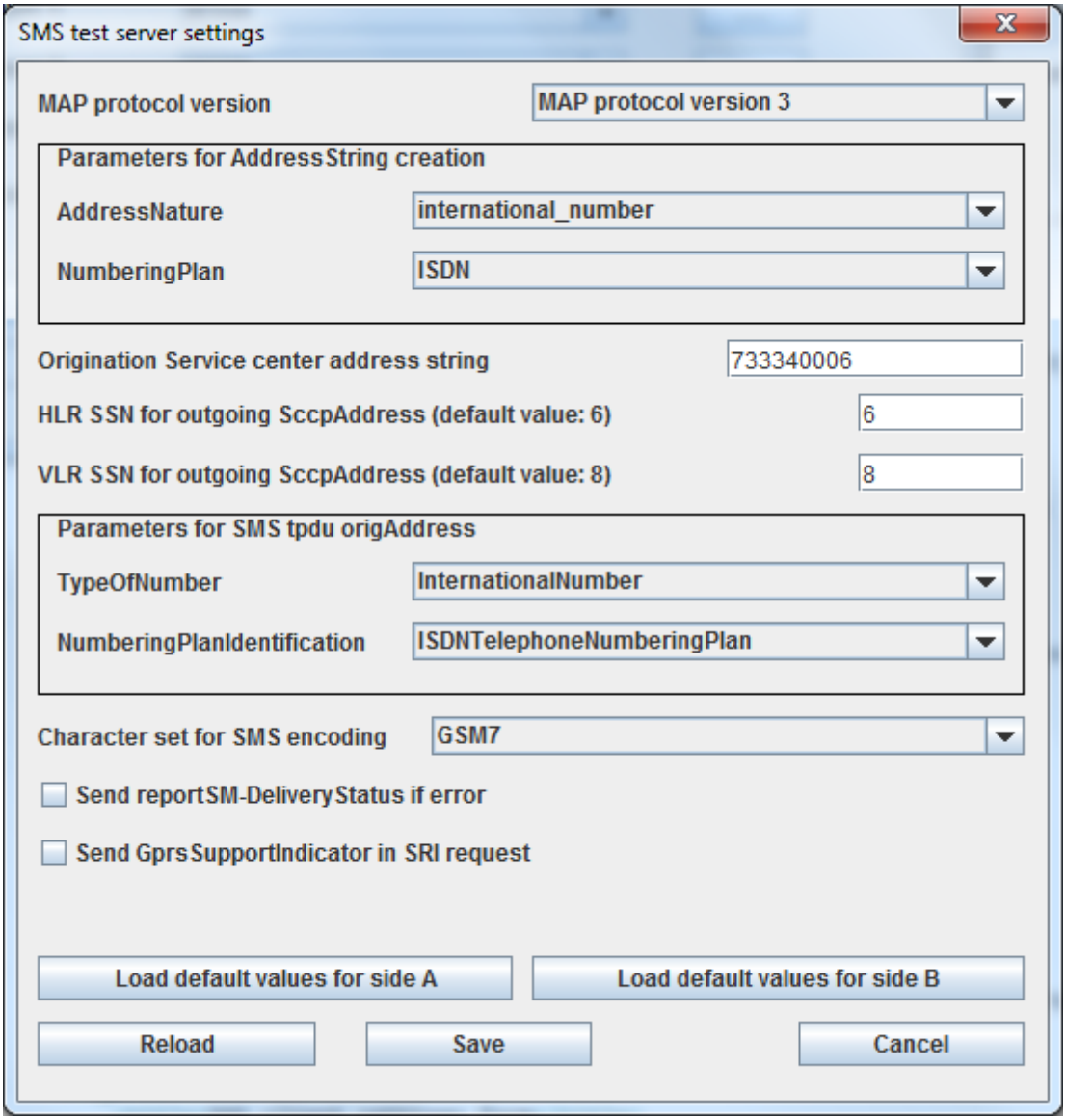


Figure 40. SMS client settings form

After starting SMS server you can:

- For sending sendRoutingInfoForSM ("Send SRIForSM" button) we should set "Destination ISDN number"
- For sending sendRoutingInfoForSM and then mt-forwardSM ("Send SRIForSM + MtForwardSM" button) we should set "Message text", "Destination ISDN number" and "Origination ISDN number"
- For sending mt-forwardSM only ("Send MtForwardSM" button) we should set "Message text", "IMSI", "VLR number" and "Origination ISDN number"

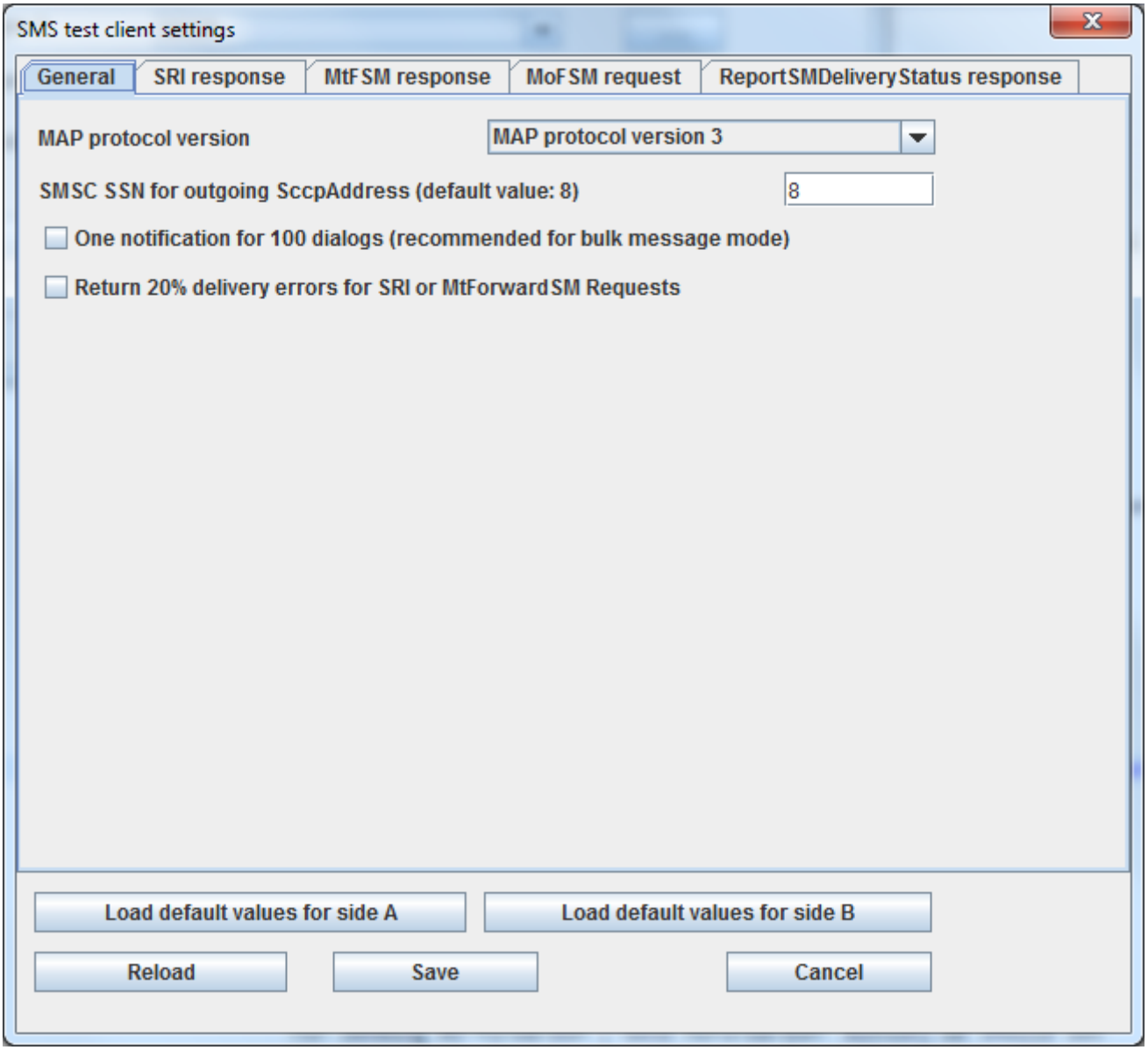
14.2.4. SMS Client

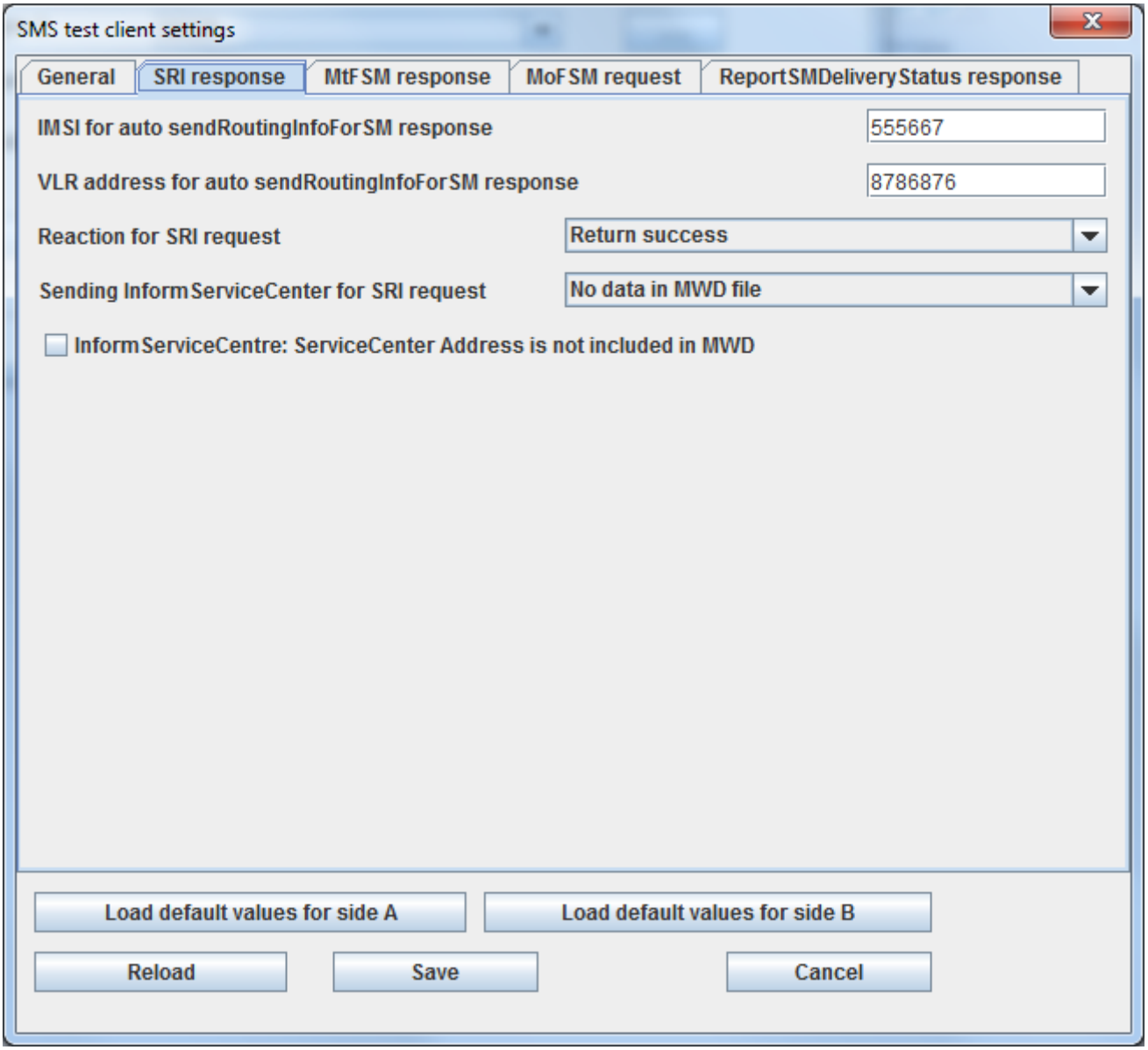
SMS Client simulates the behavior of HLR or VLR, it can:

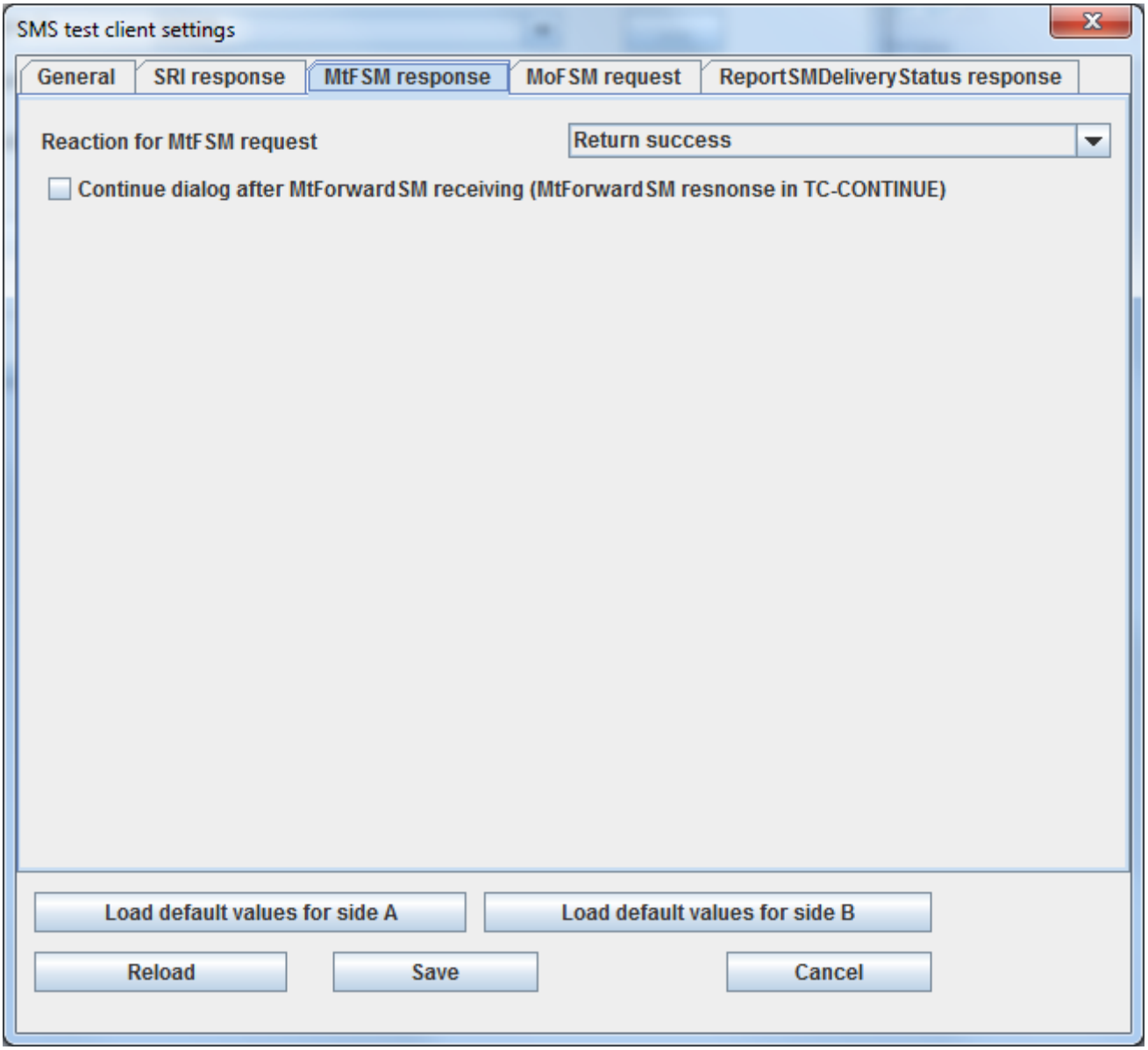
- Receive sendRoutingInfoForSM from SMSC and send as a response predefined IMSI and VLR number
- Receive mt-forwardSM from SMSC and display the received message
- Send mo-forwardSM to the SMSC

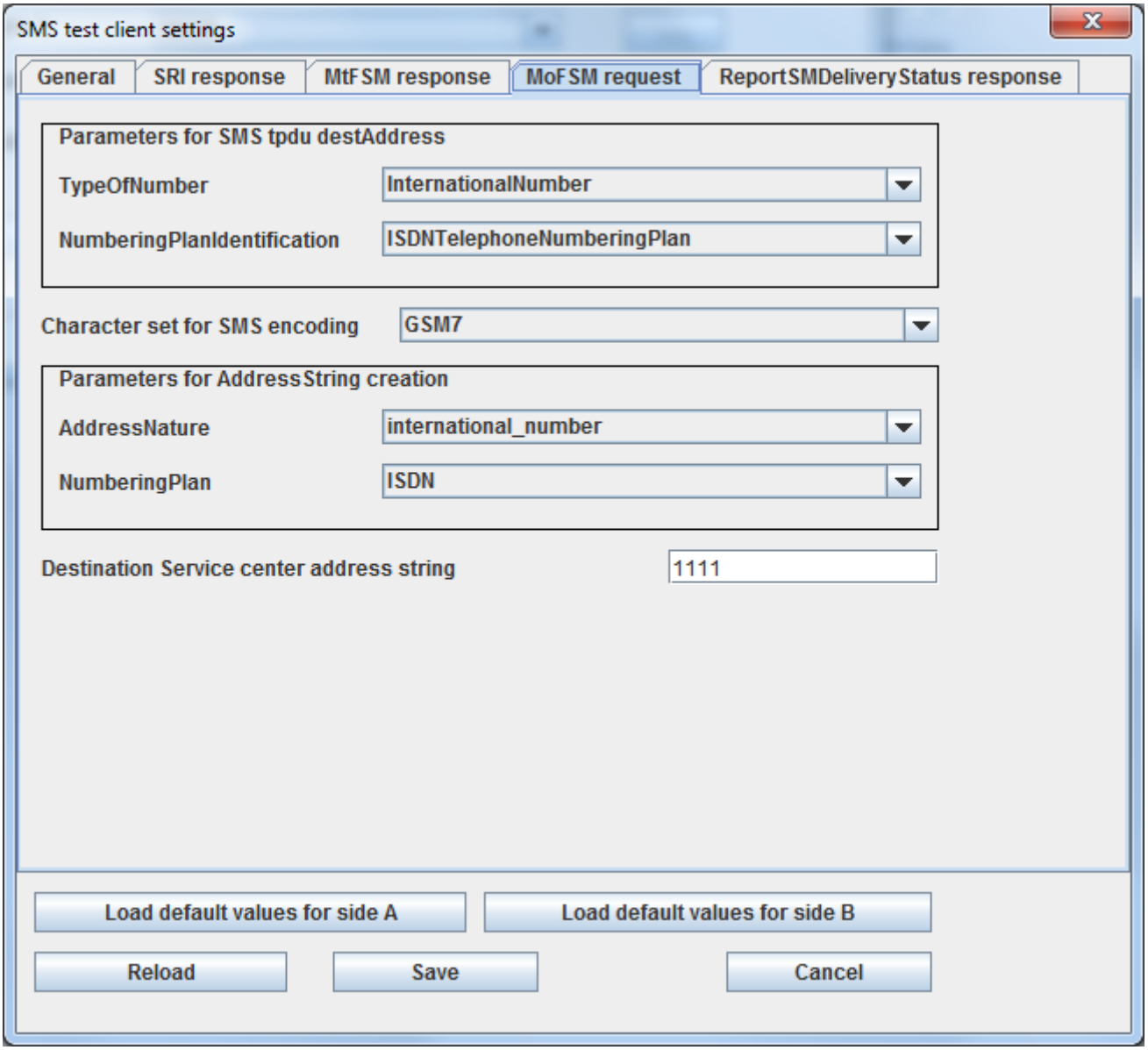
We need to set the following options to SMS client:

- AddressNature and NumberingPlan for AddressString creation
- TypeOfNumber and NumberingPlanIdentification SMS tpdu addresses
- MAP protocol version (1, 2 or 3) (version 3 is default)
- Character set for SMS message encoding (GSM7 or UCS2)
- Destination Service center address string (this address must be equal the SCCP layer CallingPartyAddress digits)
- SMSC SSN value (default value is 8)
- IMSI and VLR addresses values that will be used when responding on sendRoutingInfoForSM


SMS test client settings
General
SRI response
MtFSM response
MoFSM request
ReportSMDeliveryStatus response
MAP protocol version
MAP protocol version 3
SMSC SSN for outgoing SccpAddress (default value: 8)
8
One notification for 100 dialogs (recommended for bulk message mode)
Return 20% delivery errors for SRI or MtForwardSM Requests
Load default values for side A
Load default values for side B
Reload
Save
Cancel


SMS test client settings
General
SRI response
MtFSM response
MoFSM request
ReportSMDeliveryStatus response
IMSI for auto sendRoutingInfoForSM response
555667
VLR address for auto sendRoutingInfoForSM response
8786876
Reaction for SRI request
Return success
Sending InformServiceCenter for SRI request
No data in MWD file
InformServiceCentre: ServiceCenter Address is not included in MWD
Load default values for side A
Load default values for side B
Reload
Save
Cancel


SMS test client settings
General
SRI response
MtFSM response
MoFSM request
ReportSMDeliveryStatus response
Reaction for MtFSM request
Return success
Continue dialog after MtForwardSM receiving (MtForwardSM resnonse in TC-CONTINUE)
Load default values for side A
Load default values for side B
Reload
Save
Cancel


SMS test client settings
General
SRI response
MtFSM response
MoFSM request
ReportSMDeliveryStatus response
Parameters for SMS tpdu destAddress
TypeOfNumber
InternationalNumber
NumberingPlanIdentification
ISDNTelephoneNumberingPlan
Character set for SMS encoding
GSM7
Parameters for AddressString creation
AddressNature
international_number
NumberingPlan
ISDN
Destination Service center address string
1111
Load default values for side A
Load default values for side B
Reload
Save
Cancel

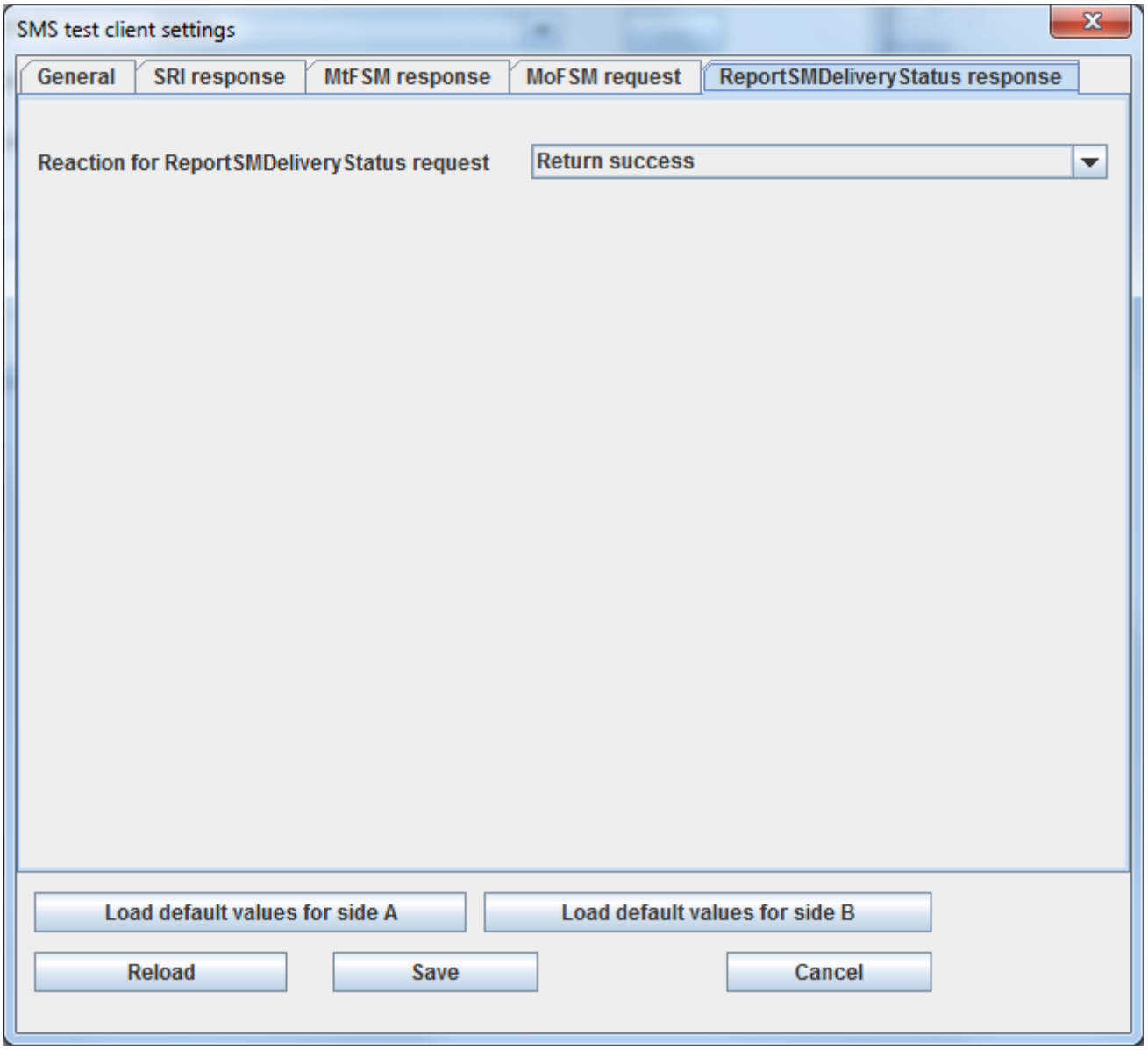


Figure 41. SMS client settings form

After starting SMS client you can:

- For sending mo-forwardSM ("Send MoForwardSM" button) we should set "Message text", "Destination ISDN number" and "Origination ISDN number"
- When recieving sendRoutingInfoForSM client automatically generate a response with preconfigured IMSI and VLR address

14.2.5. CAMEL SCF part

CAMEL SCF part simulates the behavior of CAMEL Service Control Function, it can:

- Send InitiateCallAttempt message to CAMEL SSF
- Send ApplyCharging message to CAMEL SSF
- Send Cancel message to CAMEL SSF
- Send Connect message to CAMEL SSF
- Send Continue message to CAMEL SSF

- Send ReleaseCall message to CAMEL SSF
- Send RequestReportBCSMEvent message to CAMEL SSF

We need to set the following options to CAMEL SCF:

- CAP Apllication content

Parameters of messages are hardcorded in current version and can not be configured.

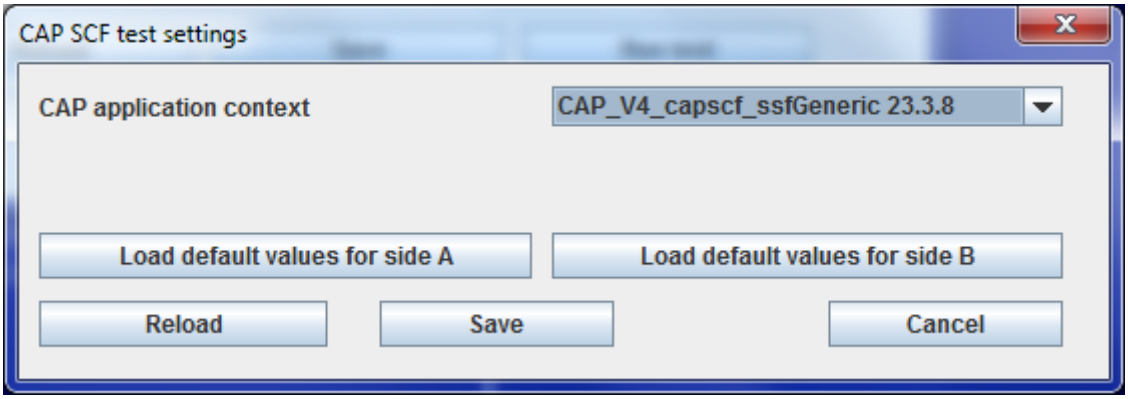


Figure 42. CAMEL SCF settings form

After CAMEL SCF server starting you can:

- For initiating of CAP dialog and sending InitiateCallAttempt - press "InitiateCallAttempt" button.
- For continuing of CAP dialog with sending ApplyCharging, Cancel, Connect, Continue, ReleaseCall or RequestReportBCSMEvent - press corresponded buttons.
- For ending of CAP dialog - press "Close Dialog" button.

14.2.6. CAMEL SSF part

CAMEL SSF part simulates the behavior of CAMEL Service Switching Function, it can:

- Send InitialDP message to CAMEL SCF
- Send AssistRequestInstructions message to CAMEL SCF
- Send ApplyChargingReport message to CAMEL SCF
- Send EventReportBCSM message to CAMEL SCF

We need to set the following options to CAMEL SSF:

- CAP Apllication content

Parameters of messages are hardcorded in current version and can not be configured.

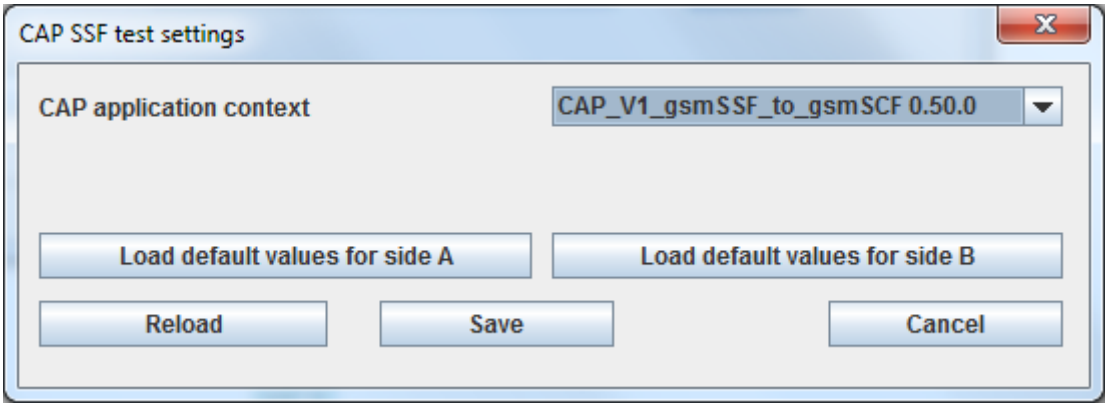


Figure 43. CAMEL SSF settings form

After CAMEL SSF server starting you can:

- For initiating of CAP dialog and sending InitialDP or AssistRequestInstructions message - press corresponded button.
- For continuing of CAP dialog with sending ApplyChargingReport or EventReportBCSM - press corresponded buttons.
- For ending of CAP dialog - press "Close Dialog" button.

14.2.7. ATI Server

ATI operation server task can be used for responding for AniTimeInterrogation MAP request. There are no configurable options for this mode now. SS7 Simulator will auto respond with ARI response with hardcoded parameters.

ATI Server test can perform followng tasks:

- Auto sending of ATI Response.

14.2.8. ATI Client

ATI Client test can perform following tasks:

- Sending a AnyTimeInterrogation MAP (ATI) request and displaying of ATI response.

For sending of ATI Request we need to specify IMSI / MSISDN digits and push "Send ATI Request" button.

Before running of ATI Client a user need to specify parameters:

- Subscriber identity type (IMSI / MSISDN (default)).
- AddressNature and NumberingPlan for AddressString Creation.
- Requested info - which parameters will be requested from ATI server.
- GSM SCF address digits.

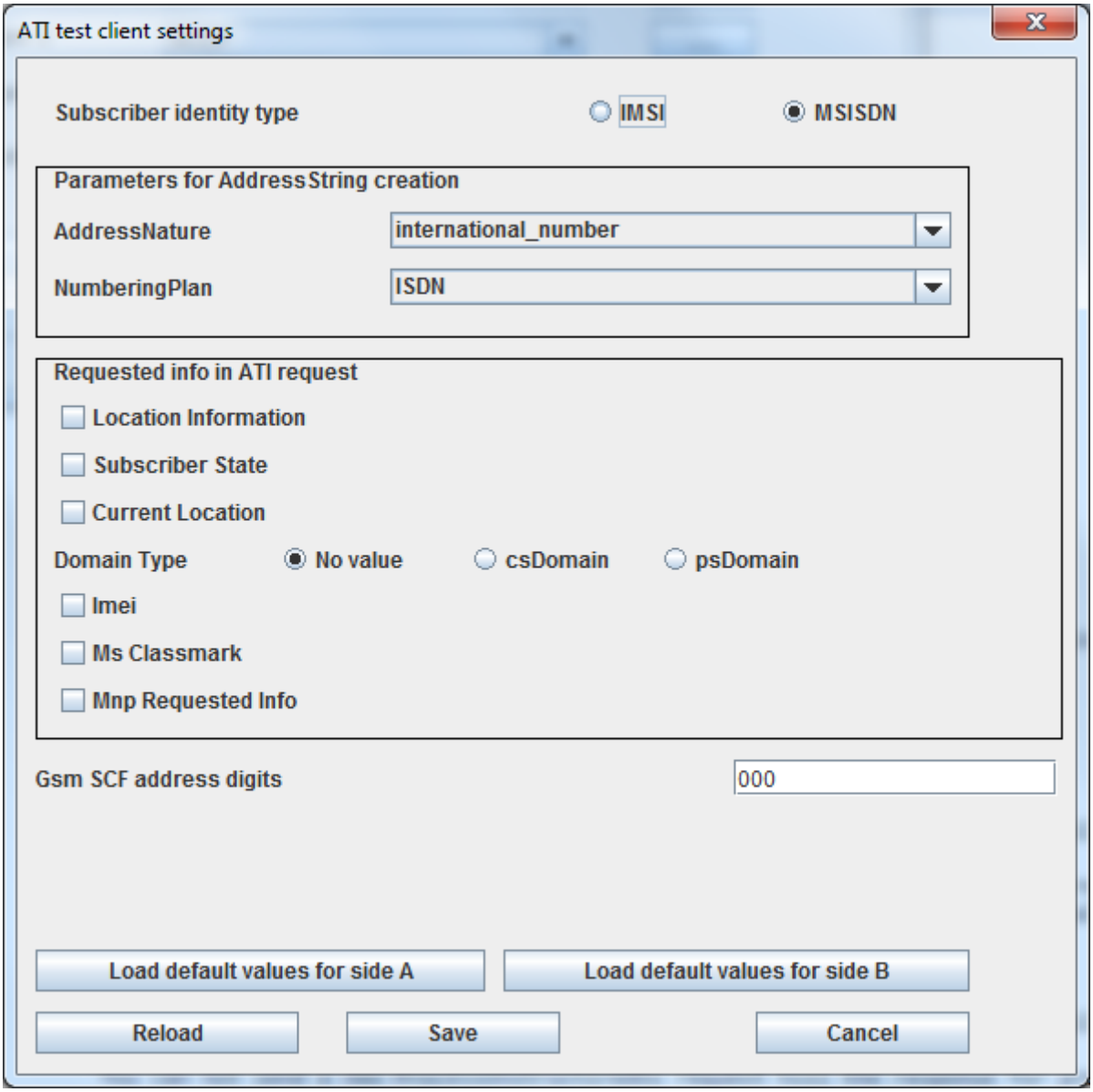


Figure 44. ATI Client settings form

Chapter 15. Trace Parser Tool

Restcomm jSS7 Trace Parser is a simple tool that can parse trace capture data files (of some formats) and log its content in some log file in a text form. This tool can be very handy to debug the flow of SS7 messages.

15.1. Running Restcomm jSS7 Trace Parser

The Restcomm jSS7 Stack comes with a Trace Parser module. You can install and run the module on any machine.

1. Change the working directory to the bin folder in the Simulator's installation directory.

```
downloads]$ cd mobicents-jss7-<version>/ss7/mobicents-ss7-traceparser/bin
```

2. (Optional) Ensure that the *run.sh* start script is executable.

```
bin$ chmod +x run.sh
```

3. Execute the *run.sh* Bourne shell script.

```
bin$ ./run.sh
```

4. Result: This will launch the GUI application as shown in the figure below:

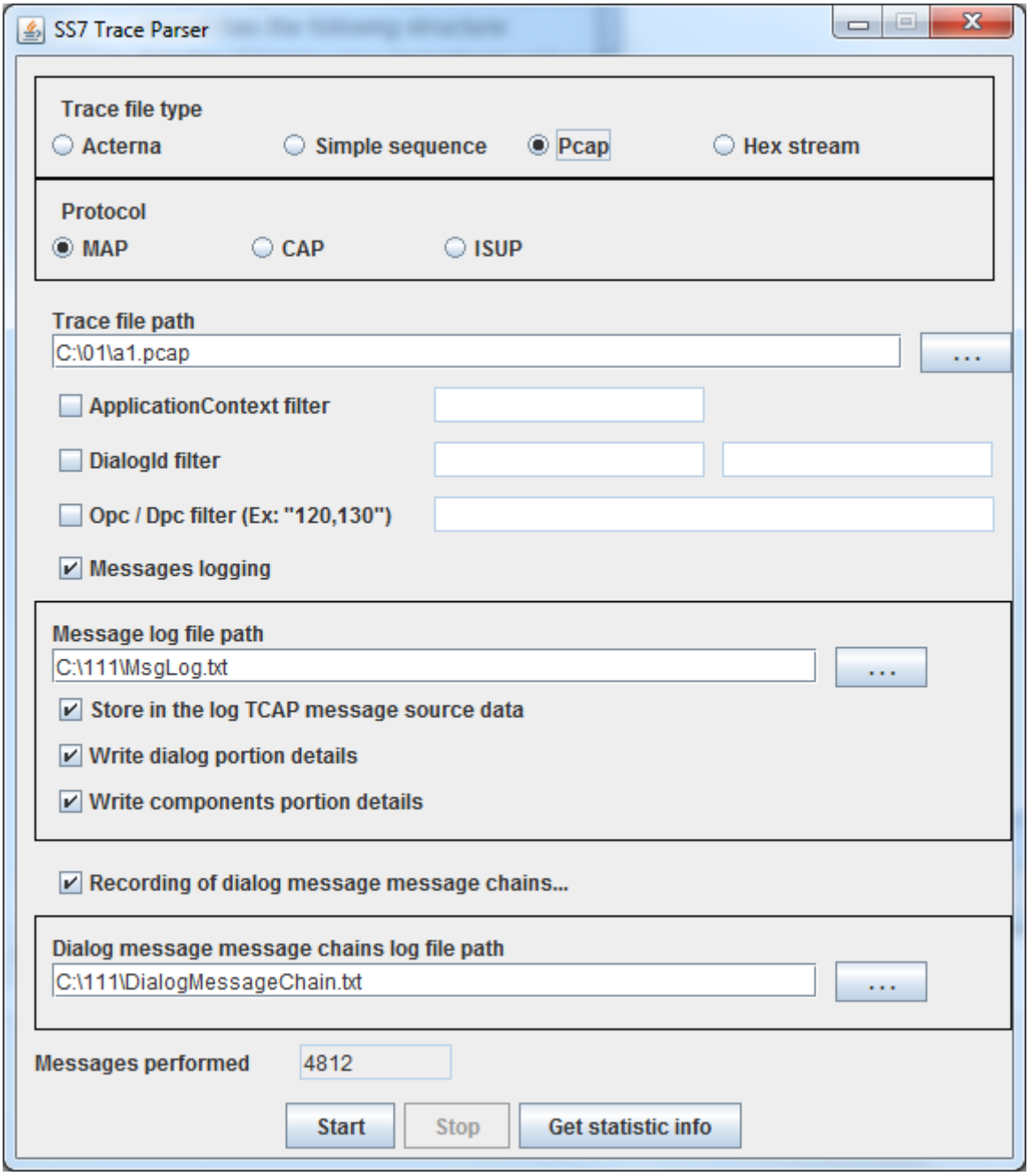


Figure 45. Running Trace Parser

15.2. Configuring Trace Parser

Once trace parser is started user can configure tool depending on requirements as explained below

- Trace file type : User can configure a trace file type for parsing. Possible values are:
  - Acterna - file format of logs of some Acterna family devices
  - Simple sequence - binary file format that content consists of several records. When you have some unsupported data format you can convert your data into this format. Each record has the following structure:
    - 2 bytes of binary encoded integer value with record length
    - without two length bytes (less significant byte is first) binary code of legacy SS7 message (Routing label and user data)

    - PCAP - pcap wireshark file format. DLT_EN10MB, DLT_LINUX_SLL and DLT_MTP3 network formats are supported now
    - Hex stream - special encoded hex text file format, it is not covered in this manaul.

- Protocol: MAP, CAP or ISUP - defines which protocol will be parsed.
- Trace file path: you have to select a file for parsing
- ApplicationContext filter - this value is used only for MAP protocol. It is integer value and used for message filtering. The value of this field is 7th number of MAP application context. For example for USSD services this value is 19. So can be filtered only all versions for a concrete application context.
- DialogId filter - for MAP and CAP protocols it is possible to filter one dialog chain. Two values in the integer form (as they are present in the text log of Trace Parser) can be used (to define origination and destination TCAP Transaction Id). Also can be only one Integer value used.
- OPC / DPC filter - this filter filters messages only for defined SS7 point codes. You can define several point codes, for example: "123,124,125". Use this filter when trace file contains a lot of data between several point codes and you want to select messages.
- Message logging - User have to define a file where text parsed data will be stored. User also needs to define which details will be saved into a log (TCAP message source data, MAP/CAP dialog portion details, components portion details)
- Recording of dialog message chains log - user can write another log that contains trace of which DPC / OPC nodes dialogs are sent. Look at log format for other details. If there are several dialogs with the same path they will be present only once in the log.

User can start parsing with the button "Start", stop with the button "Stop". After the parsing process is finished, user can get some statistic info like how many different TCAP primitives are exchanged, messages for separate Application contexts and operational codes.

Appendix A: Revision History