SS7 Stack User Guide

The guide to the Mobicents SS7 Stack

by Amit Bhayani, Bartosz Baranowski, and Oleg Kulikov

Preface v
1. Document Conventions v
1.1. Typographic Conventions v
1.2. Pull-quote Conventions vii
1.3. Notes and Warnings vii
2. Provide feedback to the authors! viii
1. Introduction to Mobicents SS7 Stack 1
1.1. Time Division Multiplexing 2
1.2. The Basics 2
1.3. Design 3
2. Installation and Running 5
2.1. Installing 5
2.1.1. Binary 5
2.2. Mobicents SS7 Service 7
2.3. Installing Mobicents SS7 Service Binary 7
2.4. Running Mobicents SS7 Service 8
2.4.1. Starting 8
2.4.2. Stopping 8
2.5. Configuring Mobicents SS7 Service 9
2.5.1. Configuring M3UA 9
2.5.2. Configuring dialogic 10
2.5.3. Configuring SCCP 11
2.5.4. Configuring ShellExecutor 12
2.5.5. Configuring SS7Service 13
2.6. Setup from source 14
2.6.1. Release Source Code Building 14
2.6.2. Development Trunk Source Building 15
3. Hardware Setup 17
3.1. Dialogic 17
4. Shell Command Line 19
4.1. Introduction 19
4.2. Starting 19
4.3. SCCP Management 20
4.3.1. Rule Management 21
4.3.2. Address Management 24
4.3.3. Remote Signaling Point Management 29
4.3.4. Remote Sub-System Management 30
4.4. M3UA Management 32
4.4.1. M3UA Management - SCTP 32
4.4.2. M3UA Management 36
5. ISUP 45
5.1. ISUP Configuration 45
5.2. ISUP Usage 47
5.3. ISUP Example 47

6. SCCP 51
6.1. Routing Management 51
6.1.1. GTT Configuration 51
6.2. SCCP Usage 55
6.3. Access Point 56
6.4. SCCP User Part Example 56
7. TCAP 59
7.1. Mobicents SS7 Stack TCAP Usage 59
7.2. Mobicents SS7 Stack TCAP User Part Example 61
8. MAP 65
8.1. SS7 Stack MAP 65
8.2. SS7 Stack MAP Usage 68
A. Java Development Kit (JDK): Installing, Configuring and Running 73
B. Setting the JBOSS_HOME Environment Variable 77
C. Revision History 81
Index 83

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or ***Proportional Bold Italic***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type `ssh` *`username@domain.name`* at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.
>
> The `mount -o remount` *`file-system`* command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.
>
> To see the version of a currently installed package, use the `rpm -q` *`package`* command. It will return a result as follows: *`package-version-release`*.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as

a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
  public static void main(String args[])
      throws Exception
  {
    InitialContext iniCtx = new InitialContext();
    Object         ref    = iniCtx.lookup("EchoBean");
    EchoHome       home   = (EchoHome) ref;
    Echo           echo   = home.create();

    System.out.println("Created Echo");

    System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
  }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the *Issue Tracker* [http://code.google.com/p/mobicents/issues/list], against the product **Mobicents SS7 Stack** , or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: SS7Stack_User_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

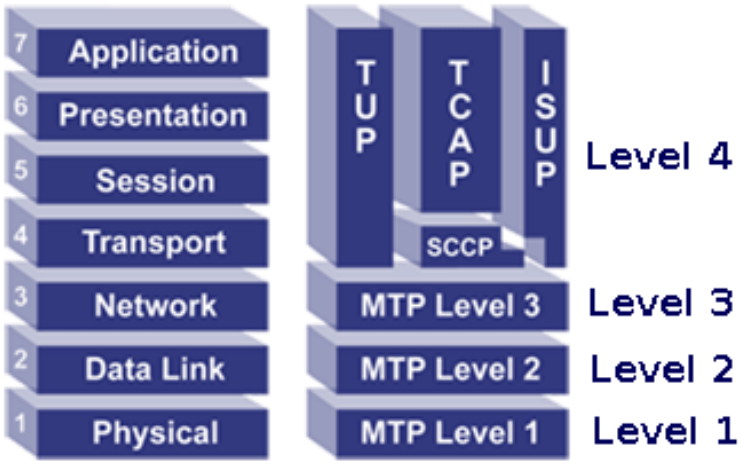
Introduction to Mobicents SS7 Stack

Important

Spaces where introduced in in some tables and code listings to ensure proper page render.

Common Channel Signaling System No. 7 (i.e., SS7 or C7) is a global standard for telecommunications defined by the *International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T)* [http://www.voip-info.org/wiki/view/ITU] . The standard defines the procedures and protocol by which network elements in the public switched telephone network (PSTN)) exchange information over a digital signaling network to effect wireless (cellular) and wireline call setup, routing and control. 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* [http://www.voip-info.org/wiki/view/ETSI]) standard used in Europe.

The hardware and software functions of the SS7 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)* [http://www.iso.ch/] .



SS7 Stack overview

Mobicents SS7 Stack is software based SS7 protocol implementation providing Level 2 and above. The Mobicents SS7 Stack is a platform in the sense that it does not provide the application itself but rather allows users to build the application

1.1. Time Division Multiplexing

In circuit switched networks such as the Public Switched Telephone Network (PSTN) there exists the need to transmit multiple subscribers' calls along the same transmission medium. To accomplish this, network designers make use of TDM. TDM allows switches to create channels, also known as tributaries, within a transmission stream. A standard DS0 voice signal has a data bit rate of 64 kbit/s, determined using Nyquist's sampling criterion. TDM takes frames of the voice signals and multiplexes them into a TDM frame which runs at a higher bandwidth. So if the TDM frame consists of n voice frames, the bandwidth will be n*64 kbit/s. Each voice sample timeslot in the TDM frame is called a channel . In European systems, TDM frames contain 30 digital voice channels, and in American systems, they contain 24 channels. Both standards also contain extra bits (or bit timeslots) for signalling (SS7) and synchronisation bits. Multiplexing more than 24 or 30 digital voice channels is called higher order multiplexing. Higher order multiplexing is accomplished by multiplexing the standard TDM frames.For example, a European 120 channel TDM frame is formed by multiplexing four standard 30 channel TDM frames.At each higher order multiplex, four TDM frames from the immediate lower order are combined, creating multiplexes with a bandwidth of n x 64 kbit/s, where n = 120, 480, 1920, etc.

1.2. The Basics

Important

Be aware, Mobicents SS7 Stack is subject to changes as it is under active development!

The Mobicents SS7 Stack is logically divided into two sections. The lower section includes SS7 Level 3 and below. The lower section is influenced by type of SS7 hardware (Level 1) used.

The upper section includes SS7 Level 4 and above. This logical division is widely based on flexibility of Mobicents SS7 Stack to allow usage of M3UA or any SS7 hardware available in the market and yet Mobicents SS7 Stack Level 4 and above remains the same.

Mobicents SS7 Stack also includes support for M3UA (SIGTRAN)

Important

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

1.3. Design

Mobicents SS7 Stack uses the abstraction layer at level 3, such that this abstraction layer exposes same API to layer 4 while it can be configured to interact with any SS7 cards via card specific module or Mobicents SS7 Stack M3UA layer.

Below diagram gives further detail on how this abstraction is achieved.

Mobicents SS7 Stack is designed such that it can be used in any container like JBoss Application Server or it can be also fired as standalone Java Application.

Installation and Running

2.1. Installing

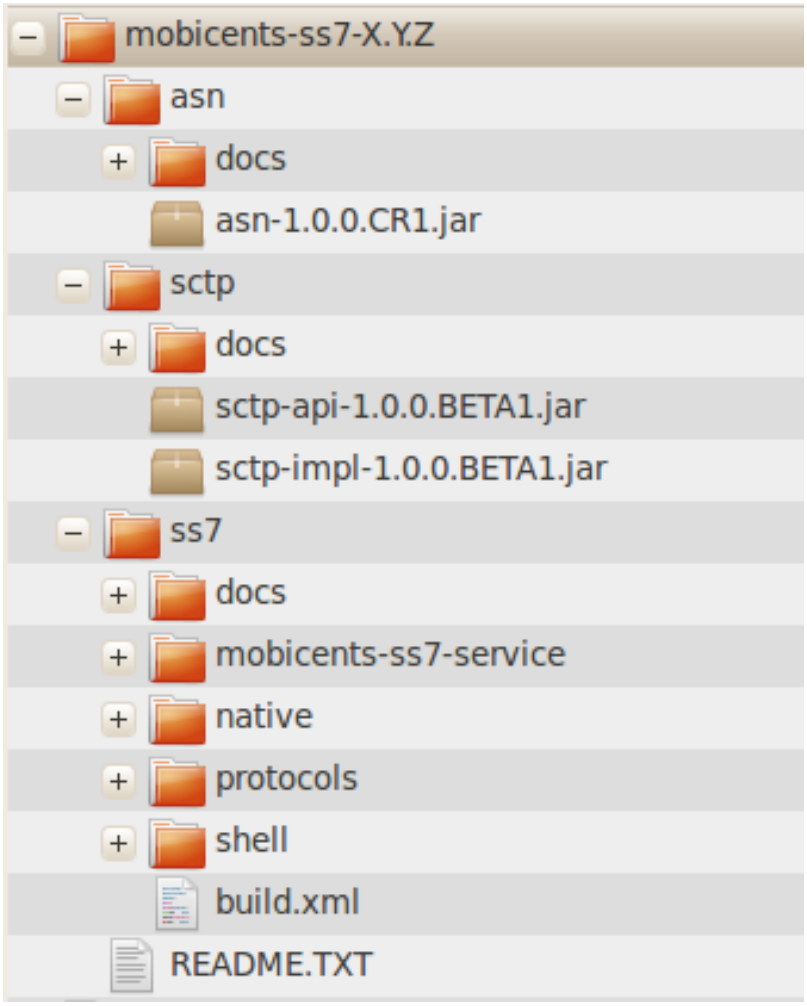
Mobicents SS7 stack at its core requires only Java if you are using only `M3UA`. However if you plan to use SS7 hardware, respective SS7 cards needs to be installed on the server along with native libraries.

A simple way to get started is to download and install binary. This will provide you with all the dependencies you need to get going. You can obtain binary release from http://sourceforge.net/projects/mobicents/files

2.1.1. Binary

The Mobicents SS7 Stack binary is broken down into a few modules.

Binary release has following layout:



Mobicents SS7 Stack binary layout.

Note

X.Y.Z in above layout is the respective release version of binary.

The following is a description of the important services and libraries that make up Mobicents SS7 Stack binary

- `asn` : Abstract Syntax Notation One (ASN.1) library is used by various Mobicents SS7 Stack protocols to encode/decode the structured data exchanged between Signaling Point over networks. To know more about asn library refer to document included with asn. Applications using any of the Mobicents SS7 Stack User Protocols may never need to call asn API directly, however it must be in classpath as Mobicents SS7 Stack User Protocols refers this library.

  To understand more about asn, refer to documentation included in `asn/docs`

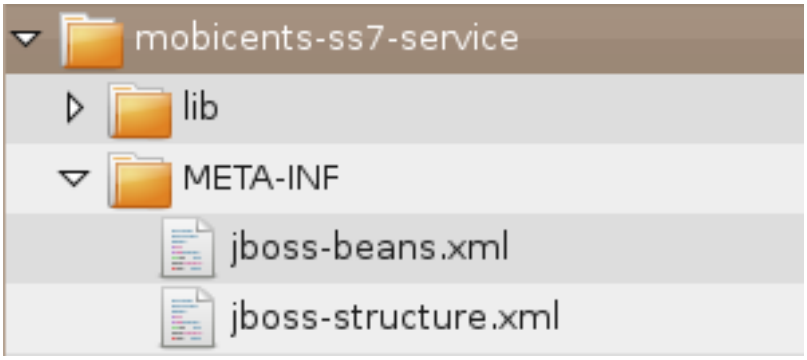
- `sctp` : Stream Control Transmission Protocol (SCTP) Library is providing the convenient API's over Java SCTP. This library will be used only if M3UA layer is used.

  To understand more about sctp, refer to documentation included in `sctp/docs`

- `ss7` : ss7 contains the core protocol libraries to be used by end application as well as service that is deployed in JBoss AS. The sub-modules included in ss7 are

  - `docs` : User guide for Mobicents SS7 Stack
  - `mobicents-ss7-service` : SS7 service is the core engine as explained in section *Section 2.2, "Mobicents SS7 Service"*

    mobicents-ss7-service binary has following layout:

  - `native` : native libraries component to interact with SS7 Card installed on server, runtime component. As of now native libraries are compiled only for linux OS. However if you plan to use `M3UA` there is no dependency on OS as everything is 100% java.
  - `protocols` : The Mobicents SS7 Stack User Protocols libraries. Your application would directly call the API's exposed by these libraries. Depending on application you may be either interested in `TCAP`, `MAP` or both or `ISUP` libraries

- `shell` : the Command Line Interface (CLI) module to manage the `mobicents-ss7-service`. Refer *Chapter 4, Shell Command Line* to understand how to use shell

2.2. Mobicents SS7 Service

As the name indicates Mobicents SS7 Service is a deployable service that can be deployed in any container that supports `JMX` and exposes `JNDI`

Mobicents SS7 Service exposes convenient way of configuring SS7 stack via CLI commands. Service wraps SS7 level 4 i.e., SCCP and lower layers and exposes it via JNDI such that layer above can do the look-up and use it in any application. The advantage of doing so is SCCP and lower layers remains same while above layers can register it-self based on Sub System Number (SSN).

2.3. Installing Mobicents SS7 Service Binary

The upper layers of Mobicents SS7 viz., TCAP, MAP depends on Mobicents SS7 Service and Mobicents SS7 Service must be installed before upper layers can be used. The Mobicents SS7 Service binary requires that you have JBoss Application Server installed and JBOSS_HOME system property set. To know further details on setting JBOSS_HOME look *Appendix B, Setting the JBOSS_HOME Environment Variable*

Once JBOSS_HOME is properly set, use ant to deploy the mobicents-ss7-service, shell scripts and shell library.

Important

Ant 1.6 (or higher) is used to install the binary. Instructions for using Ant, including install, can be found at *http://ant.apache.org/*

```
[usr]$ cd ss7-1.0.0.CR3/ss7
[usr]$ ant deploy
```

To undeploy these services

```
[usr]$ cd ss7-1.0.0.CR3/ss7
[usr]$ ant undeploy
```

While above steps will deploy the necessary ss7 service and shell components, the `java.library.path` should be set to point the directory containing native component or should be copied to JBoss native library path manually. This step is only required if you are using the SS7 board on server.

2.4. Running Mobicents SS7 Service

Starting or stopping Mobicents SS7 Service is no different than starting or stopping JBoss Application Server

2.4.1. Starting

Once installed, you can run server by executing the run.sh (Unix) or run.bat (Microsoft Windows) startup scripts in the `<install_directory>/bin` directory (on Unix or Windows). If the service started properly you should see following lines in the Unix terminal or Command Prompt depending on your environment:

```
23:22:26,079 INFO  [LinksetManager] SS7 configuration file path /
home/abhayani/workarea/mobicents/jboss-5.1.0.GA/server/default/data/
linksetmanager.xml
23:22:26,141 INFO  [LinksetManager] Started LinksetManager
23:22:26,199 INFO  [SS7Service] Starting SCCP stack...
23:22:26,229 INFO  [SccpStackImpl] Starting ...
23:22:26,230 INFO  [RouterImpl] SCCP Router configuration file: /home/
abhayani/workarea/mobicents/jboss-5.1.0.GA/server/default/deploy/mobicents-
ss7-service/sccp-routing.txt
23:22:26,261 INFO  [SS7Service] SCCP stack Started. SccpProvider bound to
 java:/mobicents/ss7/sccp
23:22:26,261 INFO  [ShellExecutor] Starting SS7 management shell environment
23:22:26,270 INFO  [ShellExecutor] ShellExecutor listening
 at /127.0.0.1:3435
23:22:26,270 INFO  [SS7Service] [[[[[[[[[ Mobicents SS7 service
 started ]]]]]]]]]
```

If you have started ss7-1.0.0.CR3 for the first time, `SS7` is not configured. You need to use Shell Client to connect to ss7-1.0.0.CR3 as defined in *Chapter 4, Shell Command Line* . With `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.

Once the configured, the state and configuration of `SS7` is persisted which stands server re-start.

2.4.2. Stopping

You can shut down the server(s) by executing the `shutdown.sh -s` (Unix) or `shutdown.bat -s` (Microsoft Windows) scripts in the `<install_directory>/bin` directory (on Unix or Windows).

Note that if you properly stop the server, you will see the following three lines as the last output in the Unix terminal or Command Prompt:

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

2.5. Configuring Mobicents SS7 Service

Configuration is done through an XML descriptor named `jboss-beans.xml` and is located at `$JBOSS_HOME/server/profile_name/deploy/mobicents-ss7-service/META-INF`, where **profile_name** is the server profile name.

The Mobicents SS7 Layer 4 (`SCCP, ISUP`) leverages either of following `MTP` layers to exchange signalling messages with remote signalling points

- `M3UA`
- `dialogic`

The ss7 service will be configured with either of these services.

2.5.1. Configuring M3UA

`M3UAManagement` is only needed if the underlying SS7 service will leverage `M3UA`. `M3UAManagement` configuration is further explained in *???*

```
                                                                        <!--
========================================================================== -->
  <!-- SCTP                                                 -->
  <!-- SCTP Management is used by M3UA only. Comment out if you are using   -->
  <!-- SS7 hardware                                         -->
                                                                        <!--
========================================================================== -->
  <bean name="SCTPManagement" class="org.mobicents.protocols.sctp.ManagementImpl">
    <constructor>
      <parameter>SCTPManagement</parameter>
    </constructor>
    <property name="persistDir">${jboss.server.data.dir}</property>
    <property name="singleThread">true</property>
    <property name="connectDelay">30000</property>
  </bean>

                                                                        <!--
========================================================================== -->
  <!-- M3UA                                                 -->
```

```
  <!-- M3UAManagement is managing the m3ua side commands                -->
  <!-- Comment out if you are using SS7 hardware                        -->
                                                                        <!--
=========================================================================== -->

 <bean name="Mtp3UserPart" class="org.mobicents.protocols.ss7.m3ua.impl.M3UAManagement">
     <constructor>
       <parameter>Mtp3UserPart</parameter>
     </constructor>
     <property name="persistDir">${jboss.server.data.dir}</property>
     <property name="transportManagement">
       <inject bean="SCTPManagement" />
     </property>
  </bean>

  <bean name="M3UAShellExecutor"
     class="org.mobicents.protocols.ss7.m3ua.impl.oam.M3UAShellExecutor">
     <property name="m3uaManagement">
       <inject bean="Mtp3UserPart" />
     </property>
     <property name="sctpManagement">
       <inject bean="SCTPManagement" />
     </property>
  </bean>
```

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

For example in above case, when Mobicents SS7 Service is started file `SCTPManagement_sctp.xml` will be created at `$JBOSS_HOME/server/profile_name/data` directory

`org.mobicents.protocols.ss7.m3ua.impl.M3UAManagement` takes `String` as constructor argument. The name is prepend to `xml` file created by M3UA stack for persisting state of M3UA resources. The `xml` is stored in path specified by `persistDir` property above.

For example in above case, when Mobicents SS7 Service is started file `Mtp3UserPart_m3ua.xml` will be created at `$JBOSS_HOME/server/profile_name/data` directory

2.5.2. Configuring dialogic

`Dialogic` based MTP layer will only be used when you have installed Dialogic cards. `DialogicMtp3UserPart` communicates with Dialogic hardware. Its asumed here that MTP3 and MTP2 is leveraged from Dialogic stack either on-board or on-host.

```
                                                                                <!--
========================================================================== -->
  <!-- Dialogic Mtp3UserPart -->
                                                                                <!--
========================================================================== -->

 <bean name="Mtp3UserPart" class="org.mobicents.ss7.hardware.dialogic.DialogicMtp3UserPart">
                                                    <property      name="sourceModuleId">61</
property> <property name="destinationModuleId">34</property>
      </bean>
```

`sourceModuleId` is source module id and should match with one configured in system.txt used by dialogic drivers. Here 61 is assigned for Mobicents process. `destinationModuleId` is destination module id. 34 is default Dialogic MTP3 module id.

2.5.3. 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="localSpc">6535</property>
    <property name="ni">3</property>
    <property name="persistDir">${jboss.server.data.dir}</property>
    <property name="removeSpc">true</property>
    <property name="mtp3UserPart">
      <inject bean="Mtp3UserPart" />
    </property>
  </bean>

  <bean name="SccpExecutor"
    class="org.mobicents.protocols.ss7.sccp.impl.oam.SccpExecutor">
    <property name="sccpStack">
      <inject bean="SccpStack" />
```

```
    </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 Mobicents SS7 Service is started two file's `SccpStack_sccpresource.xml` and `SccpStack_sccprouter.xml` will be created at `$JBOSS_HOME/server/profile_name/data` directory

Stack has following properties:

localSpc
: property specifies the local signaling point code.

ni
: specifies the network indicator that forms the part of service information octet (SIO)

persistDir
: As explained above

removeSpc
: After Global Title Translation, if the SCCP address includes the destination point code (DPC) however Address Indicator (AI) indicates route on Global Title and `removeSpc` is set to true, DPC will be removed from SCCP Address. The same rule applies for both calling as well as called party SCCP Address.

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

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

2.5.4. Configuring ShellExecutor

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

```
<!-- ============================================= -->
  <!-- Shell Service                                 -->
  <!-- ============================================= -->
  <!-- Define Shell Executor -->
  <bean name="ShellExecutor" class="org.mobicents.ss7.ShellExecutor">
    <property name="address">${jboss.bind.address}</property>
```

```
    <property name="port">3435</property>

    <property name="sccpExecutor">
      <inject bean="SccpExecutor" />
    </property>
    <property name="m3UAShellExecutor">
      <inject bean="M3UAShellExecutor" />
    </property>
  </bean>
```

By default ShellExecutor listens at `jboss.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.

Note

To understand JBoss bind options look at *Installation_And_Getting_Started_Guide* [http://docs.jboss.org/jbossas/docs/Installation_And_Getting_Started_Guide/5/html_single/index.html]

2.5.5. Configuring SS7Service

`SS7Service` acts as core engine binding all the components together.

```
                                                                    <!--
================================================================== -->
  <!-- Mobicents SS7 Service -->
                                                                    <!--
================================================================== -->
  <bean name="SS7Service" class="org.mobicents.ss7.SS7Service">
    <annotation>@org.jboss.aop.microcontainer.aspects.jmx.JMX(name=
        "org.mobicents.ss7:service=SS7Service",
        exposedInterface=org.mobicents.ss7.SS7ServiceMBean.class,
        registerDirectly=true)</annotation>
    <property name="jndiName">java:/mobicents/ss7/sccp</property>
    <property name="stack">
      <inject bean="SccpStack" property="sccpProvider" />
    </property>
  </bean>
```

SS7 service binds SccpProvider to JNDI `java:/mobicents/ss7/sccp`. The JNDI name can be configured to any valid JNDI name specific to your application.

2.6. Setup from source

Mobicents SS7 Stack is an open source project, instructions for building from source are part of the manual! Building from source means you can stay on top with the latest features. Whilst aspects of Mobicents SS7 Stack are quite complicated, you may find ways to become contributors.

Mobicents SS7 Stack works with JDK1.5 and above (If using M3UA, JDK1.7 and above). you will also need to have the following tools installed. Minimum requirement version numbers provided.

- `Subversion Client 1.4` : Instructions for using SVN, including install, can be found at *http://subversion.tigris.org*
- `Maven 2.0.9` : Instructions for using Maven, including install, can be found at *http://maven.apache.org/*
- `Ant 1.7.0` : Instructions for using Ant, including install, can be found at *http://ant.apache.org*

2.6.1. Release Source Code Building

1. **Downloading the source code**

   Use SVN to checkout a specific release source, the base URL is http://mobicents.googlecode.com/svn/tags/protocols/ss7, then add the specific release version, lets consider 1.0.0.CR3.

```
[usr]$ svn co http://mobicents.googlecode.com/svn/tags/protocols/ss7/ss7-1.0.0.CR3
```

2. **Building the source code**

   Now that we have the source the next step is to build and install the source. Mobicents SS7 Stack uses Maven 2 to build the system. There are two profiles. Default one builds only java source. The other profile "dialogiclinux" additionaly compile native modules.

Note

Native modules are supported only for linux OS for now.

   Use "dialogiclinux" profile if linux server on which this code is built already has dialogic module installed. Make sure you pass "include.dialogic" and "include.dialogic.gctlib" system property pointing to correct directory where dialogic libraries are installed. include.dialogic.gctlib points to directory where `gctload` is present (generally /opt/dpklnx for linux OS)

```
```

```
[usr]$ cd ss7-1.0.0.CR3
[usr]$ mvn install -Pdialogclinux -Dinclude.dialogic=/opt/dpklnx/INC -Dinclude.dialogic.gctlib=/opt/dpklnx
```

To build Mobicents SS7 Stack without building any native libraries use

```
[usr]$ cd ss7-1.0.0.CR3
[usr]$ mvn install
```

Note

If you are using Mobicents SS7 Stack without any native dependencies, Mobicents SS7 Stack can run on any OS.

Use Ant to build the binary .

```
[usr]$ cd ss7-1.0.0.CR3/release
[usr]$ ant
```

2.6.2. Development Trunk Source Building

Similar process as for *Section 2.6.1, "Release Source Code Building"*, the only change is the SVN source code URL, which is http://mobicents.googlecode.com/svn/trunk/protocols/ss7.

Hardware Setup

This chapter contains reference to configure hardware drivers for different types of `SS7` cards.

Mobicents SS7 Stack supports `dialogic` based SS7 cards which has on board support for MTP2/ MTP3

3.1. Dialogic

To install Dialogic cards visit the Dialogic site at *http://www.dialogic.com/*

Shell Command Line

4.1. Introduction

Mobicents SS7 Stack provides Shell client to manage configuration of SS7 Stack Services. This chapter describes how to install and start client. Also it describes available commands and provides examples. To see examples of specific flow, to perform certain tasks, please refer to sections in chapter devoted to `SCCP` or `M3UA` .

4.2. Starting

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

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

Once console starts, it will print following information:

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

Mobicents SS7: release.version=1.0.0-SNAPSHOT
This is free software, with components licensed under the GNU General Public
 License
version 2 and other licenses. For further details visit http://mobicents.org
=========================================================================

mobicents>
```

The `ss7-run` script supports following options

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

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

```
ss7 connect <IP> <PORT>
```

Example 4.1. Connec to remote machine

```
mobicents>ss7 connect 10.65.208.215 3435

mobicents(10.65.208.215:3435)>
```

Note

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

Command to disconnect has following structure:

```
ss7 discconnect
```

Example 4.2. Disconnect

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

Bye
mobicents>
```

4.3. 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. SCCP is used as the transport layer for TCAP -based services

As SCCP acts as message router, it requires means to configure routing information. CLI provides way to easily manage routing rules information in Mobicents SCCP implementation.

User should also configure the remote subsystem number and remote signaling pointcode. In some cases where global title is used, SCCP will only require configuring of remote signaling pointcode and configuring of remote subsystem is not required.

4.3.1. Rule Management

SCCP routing rules are managed by `sccp rule` command. It allows to perform following:

- `sccp rule create`
- `sccp rule modify`
- `sccp rule delete`
- `sccp rule show`

4.3.1.1. Create Rule

Rule can be create by issuing command with following structure:

```
sccp rule create <id> <mask> <address-indicator> <point-code> <subsystem-number>
 <translation-type> <numbering-plan>
<nature-of-address-indicator> <digits> <primary-address-id> <backup-address-id>
```

This command should be specified after `primary_add` and `backup_add` are configured. Please refer *Section 4.3.2, "Address Management"* on how to configure `primary_add` and `backup_add`

<id>
A unique number to identify this rule

<mask>
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. mask is divided into sections by separator /. The number of sections in mask should be equal to sections in digits passed in this command and sections in primary or backup address

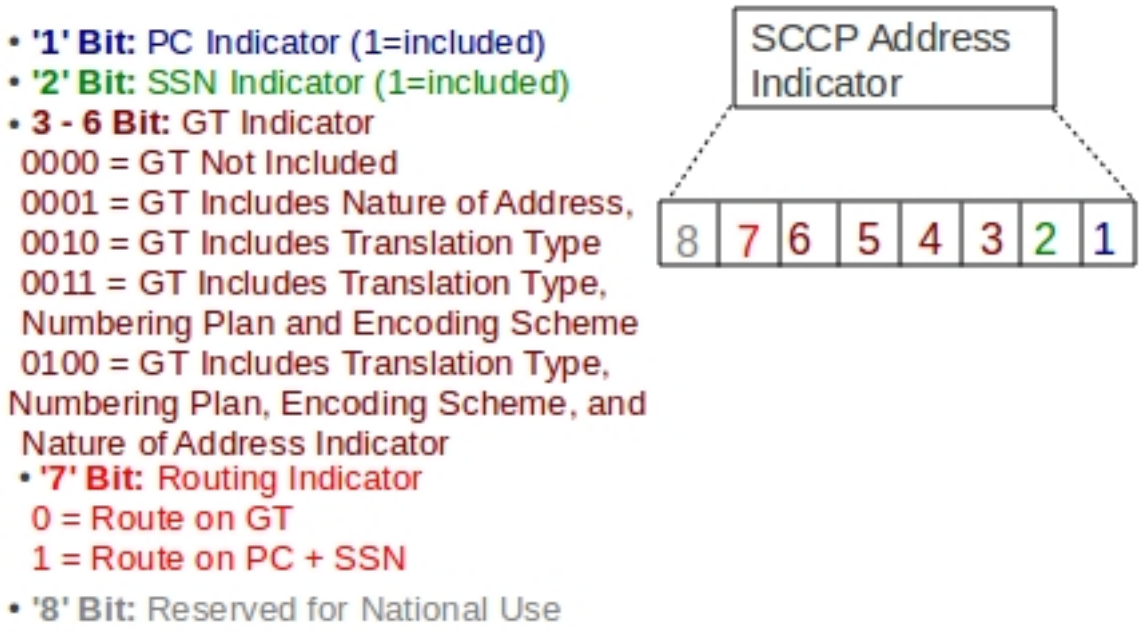
Table 4.1. mask definitions

Mnemonic	Function
-	Ignore
/	Separator used to split the mask into sections.
K	Keep the original dialed digits of this

Mnemonic	Function
	section into translated digits
R	Replace the original dialed digits of this section with same section from primary or backup address into translated digits

<address-indicator>

The address indicator is the first field in 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 original global title is then compared with passed address information to match the rule.



SCCP Address Indicator

<point-code>

Point code. This is ignored if bit 0 of address-indicator is not set.

<subsystem-number>

Subsystem Number. This is ignored if bit 1 of address-indicator is not set.

<translation-type>

Translation type. This is ignored if GT Indicator is 0000 or 0001

Table 4.2. Translation Type Values

Value	Description
0	Unknown
1 to 63	International Service
64 to 127	Spare
128 to 254	National Network Specific
255	Reserved for Expansion

<numbering-plan>
The Number Plan (NP) field specifies the numbering plan that the address information follows. This is ignored if GT Indicator is 0000, 0001 or 0010

<nature-of-address-indicator>
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

<digits>
Specifies the string of digits divided into subsections using separator '/' depending on if mask contains separator. The dialed digits should match with theses digits as per rule specified bellow

Table 4.3. digit pattern

Value	Description
-	padding - ignored
*	wildcard - matches any number of digits
?	wildcard - matches exactly one digit
/	sparator used to split the digit pattern into sections. Each section can be processed differently as specified by mask parameter.

<primary-address-id>
Identifies the SCCP Address used as the primary translation

<backup-address-id>
Identifies the SCCP Address used as the backup translation incase if pointcode specified by primary address is not available

Example 4.3. SCCP Rule creation

```
mobicents(10.65.208.215:3435)>sccp rule create 1 R 71 2 8 0 0 3 123456789 1
mobicents(10.65.208.215:3435)>sccp rule create 2 R 71 2 8 0 0 3 123456789 1
 1
```

4.3.1.2. Delete SCCP Rule

SCCP Rule can be deleted by issuing command with following structure:

```
sccp rule delete  <id>
```

Where:

<id>
is id set during rule creation

Example 4.4. SCCP Rule Removal

```
mobicents(10.65.208.215:3435)>sccp rule delete 1
Rule successfully removed
```

4.3.1.3. Show SCCP Rule

Rule's can be viewed by issuing command with following structure:

```
sccp rule show <id>
```

Where:

<id>
id is optional. If passed only rule matching the id will be shown, else all the rules will be shown

4.3.2. Address Management

The command is used to define primary or backup address of translation. 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.The syntanx remains same except for primary address `sccp primary_add` is used and for backup address `sccp backup_add` is used

- `sccp primary_add create`

  `sccp backup_add create`
- `sccp primary_add modify`

  `sccp backup_add modify`
- `sccp primary_add delete`

  `sccp backup_add delete`
- `sccp primary_add show`

  `sccp backup_add show`

4.3.2.1. Create Address

Address can be create by issuing command with following structure:

- For primary address

```
sccp primary_add create <id> <address-indicator> <point-code> <subsystem-number>
 <translation-type> <numbering-plan>
<nature-of-address-indicator> <digits>
```

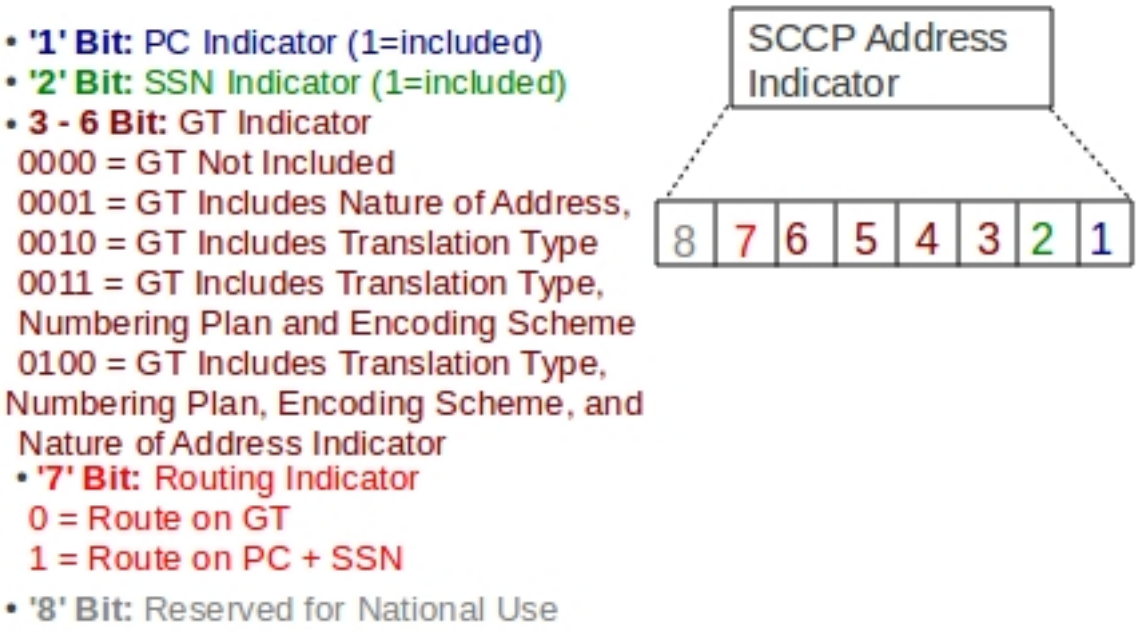
- For backup address

```
sccp backup_add create <id> <address-indicator> <point-code> <subsystem-number>
 <translation-type> <numbering-plan>
<nature-of-address-indicator> <digits>
```

<id>
: A unique number to identify this address

<address-indicator>
: The address indicator is the first field in 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 original global title is then compared with passed address information to match the rule.



SCCP Address Indicator

<point-code>

Point code. This is ignored if bit 0 of address-indicator is not set.

<subsystem-number>

Subsystem Number. This is ignored if bit 1 of address-indicator is not set.

<translation-type>

Translation type. This is ignored if GT Indicator is 0000 or 0001

Table 4.4. Translation Type Values

Value	Description
0	Unknown
1 to 63	International Service
64 to 127	Spare
128 to 254	National Network Specific
255	Reserved for Expansion

<numbering-plan>

The Number Plan (NP) field specifies the numbering plan that the address information follows. This is ignored if GT Indicator is 0000, 0001 or 0010

<nature-of-address-indicator>
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

<digits>
The global title address information to translate to, specified as string of digits divided into subsections using separator '/' depending on if mask contains separator.

In addition the digits string can contain

Table 4.5. Address digit

Value	Description
-	padding - ignore
/	Separtor to split the digits into sections. Each section is processed differently as specified by the mask in `sccp rule create` command.

Example 4.5. SCCP Primary Address creation

```
mobicents(10.65.208.215:3435)>sccp primary_add create 1 71 2 8 0 0 3
 123456789
```

Example 4.6. SCCP Backup Address creation

```
mobicents(10.65.208.215:3435)>sccp backup_add create 1 71 3 8 0 0 3
 123456789
```

4.3.2.2. Delete Address

- For primary address

```
sccp primary_add delete <id>
```

- For backup address

```
sccp backup_add delete <id>
```

Where:

<id>
is id set during address creation

Example 4.7. Primary Address Removal

```
mobicents(10.65.208.215:3435)>sccp primary_add delete 1
Rule successfully removed
```

Example 4.8. Backup Address Removal

```
mobicents(10.65.208.215:3435)>sccp backup_add delete 1
Rule successfully removed
```

4.3.2.3. Show Address

Address's can be viewed by issuing command with following structure:

- For primary address

```
sccp primary_add show <id>
```

- For backup address

```
sccp backup_add show <id>
```

Where:

<id>
id is optional. If passed only address matching the id will be shown, else all the addresses will be shown

4.3.3. Remote Signaling Point Management

SCCP resources includes remote signaling point and remote subsytem. Each remote signaling point that SCCP can communicate with must be configured using `sccp rsp` command

- `sccp rsp create`
- `sccp rsp modify`
- `sccp rsp delete`
- `sccp rsp show`

4.3.3.1. Create Remote Signaling Point

Remote signaling point can be create by issuing command with following structure:

```
sccp rsp create <id> <remote-spc> <rspc-flag> <mask>
```

<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

Example 4.9. Remote Signalin Point creation

```
mobicents(10.65.208.215:3435)>sccp rsp create 1 6477 0 0
```

4.3.3.2. Delete Remote Signaling Point

```
sccp rsp delete <id>
```

Where:

<id>
is id set during remote signaling point creation

Example 4.10. Remote Signaling Point removal

```
mobicents(10.65.208.215:3435)>sccp rsp delete 1
```

4.3.3.3. Show Remote Signaling Point/s

Remote signaling point can be viewed by issuing command with following structure:

```
sccp rsp show <id>
```

Where:

<id>
id is optional. If passed only remote signaling point matching the id will be shown, else all the addresses will be shown

4.3.4. Remote Sub-System Management

SCCP resources includes remote signaling point and remote subsytem. Each remote subsystem that SCCP can communicate with must be configured using `sccp rss` command

- `sccp rss create`
- `sccp rss modify`
- `sccp rss delete`
- `sccp rss show`

This command should be specified after remote signaling point is configured. Please refer *Section 4.3.3, "Remote Signaling Point Management"* on how to configure remote signaling point

4.3.4.1. Create Remote Sub-System

Remote subsystem can be created by issuing command with following structure:

```
```

```
sccp rss create <id> <remote-spc> <remote-ssn> <rss-flag>
```

<id>
A unique number to identify this remote subsystem

<remote-spc>
The remote signaling point where this remote susbsytem is deployed

<remote-ssn>
The remote subsystem number

<rss-flag>
32 bit value. Not used for now. Reserved for future

Example 4.11. Remote Sub-System creation

```
mobicents(10.65.208.215:3435)>sccp rss create 1 6477 8 0
```

4.3.4.2. Delete Remote Sub-System

```
sccp rss delete <id>
```

Where:

<id>
is id set during remote subsystem creation

Example 4.12. Remote Sub-System removal

```
mobicents(10.65.208.215:3435)>sccp rss delete 1
```

4.3.4.3. Show Remote Sub-System/s

Remote subsystem can be viewed by issuing command with following structure:

```
sccp rss show <id>
```

Where:

<id>
id is optional. If passed only remote subsystem matching the id will be shown, else all will be shown

4.4. M3UA Management

M3UA stack is also responsible to manage the SCTP Associations.

4.4.1. M3UA Management - SCTP

M3UA - SCTP is managed by `sctp` command. It allows to perform following:

- `sctp server create`
- `sctp server destroy`
- `sctp server start`
- `sctp server stop`
- `sctp server show`
- `sctp association create`
- `sctp association destroy`
- `sctp association show`

4.4.1.1. Create SCTP Server

SCTP Server can be created by issuing command with following structure:

```
sctp server create <server-name> <host-ip> <host-port>
```

Where:

server-name
Unique name assigned to the server.

host-ip
The host ip address where underlying SCTP server socket will bind

host-port
The host port where underlying SCTP server socket will bind

Example 4.13. SCTP Server creation

```
mobicents(127.0.0.1:3436)>sctp server create TestServer 127.0.0.1 2905
Successfully added Server=TestServer
```

4.4.1.2. Destroy SCTP Server

SCTP Server can be destroyed by issuing command with following structure:

```
sctp server destroy <server-name>
```

Where:

server-name
: Unique name of the server to be destroyed. Make sure server is stopped before destroying.

Example 4.14. Destroy SCTP Server

```
mobicents(127.0.0.1:3436)>sctp server destroy TestServer
Successfully removed Server=TestServer
```

4.4.1.3. Start SCTP Server

SCTP Server can be started by issuing command with following structure:

```
sctp server start <server-name>
```

Where:

server-name
: Unique name of the server to be started. The underlying SCTP server socket is bound to ip:port configured at creation time.

Example 4.15. Start SCTP Server

```
mobicents(127.0.0.1:3436)>sctp server start TestServer
```

```
Successfully started Server=TestServer
```

4.4.1.4. Stop SCTP Server

SCTP Server can be stopped by issuing command with following structure:

```
sctp server stop <server-name>
```

Where:

server-name
: Unique name of the server to be stopped. The underlying socket is closed at this point and all resource are released.

Example 4.16. Stop SCTP Server

```
mobicents(127.0.0.1:3436)>sctp server stop TestServer
Successfully stopped Server=TestServer
```

4.4.1.5. Show SCTP Server

SCTP Server's configuration can be viewed by issuing command with following structure:

```
sctp server show
```

Example 4.17. Show SCTP Server

```
mobicents(127.0.0.1:3436)>sctp server show
Not supported yet
```

4.4.1.6. Create SCTP Association

Association can be created by issuing command with following structure:

```
sctp association create <assoc-name> <CLIENT | SERVER> <server-name> <peer-ip> <peer-port> <host-ip> <host-port>
```

Where:

assoc-name
: Unique name of the association

CLIENT | SERVER
: If this association is client side or server side. If its client side, it will initiate the connection to peer and bind's to host-ip:host-port trying to connect to peer-ip:peer-port.

  If its server side, it waits for peer to initiate connection. The connection request will be accepted from peer-ip:peer-port. host-ip and host-port is not required, even if passed it will be ignored

server-name
: If this association is server side, server-name must be passed to associate with server. Server with server-name should have already been created by using command *Section 4.4.1.1, "Create SCTP Server"*

  If this association is client side, server-name shouldn't be passed.

Example 4.18. Create CLIENT SCTP Association

```
mobicents(192.168.56.1:3436)>sctp association create Assoc1 CLIENT
 192.168.56.101 2905 192.168.56.1 2905
Successfully added client Association=Assoc1
```

Example 4.19. Create SERVER SCTP Association

```
mobicents(192.168.56.1:3436)>sctp association create Assoc2 SERVER
 TestServer 192.168.56.1 2905
Successfully added server Association=TestServer
```

4.4.1.7. Destroy SCTP Association

Association can be destroyed by issuing command with following structure:

```
sctp association destroy <assoc-name>
```

Where:

assoc-name
: Unique name of the association to be destroyed

Example 4.20. Destroy SCTP Association

```
mobicents(192.168.56.1:3436)>sctp association destroy Assoc1
Successfully removed association=Assoc1
```

4.4.1.8. Show SCTP Association

Configuration of Association can be viewed by issuing command with following structure:

```
sctp association show
```

Example 4.21. Show SCTP Association

```
mobicents(192.168.56.1:3436)>sctp association show
Not supported yet
```

4.4.2. M3UA Management

M3UA is managed by `m3ua` command. It allows to perform following:

- `m3ua as create`
- `m3ua as destroy`
- `m3ua as show`
- `m3ua asp create`
- `m3ua asp destroy`
- `m3ua asp show`
- `m3ua asp start`
- `m3ua asp stop`
- `m3ua as add`
- `m3ua as remove`
- `m3ua route add`
- `m3ua route remove`
- `m3ua route show`

4.4.2.1. Create AS

Application Server (AS) can be created by issuing command with following structure:

```
m3ua as create <as-name> <AS | SGW | IPSP> mode <SE | DE> ipspType <client | server> rc
 <routing-context> traffic-mode <traffic mode>
```

Where:

as-name
: simple string name, which identifies AS. Make sure this is unique. This is mandatory parameter

AS | SGW | IPSP
: Specify if this is of type AS or SGW or IPSP. This is mandatory parameter

SE | DE
: Specify if the single or double exchange of ASP State Maintenance (ASPSM) and ASP Traffic Maintenance (ASPTM) messages should be performed. This is mandatory parameter.

client | server
: If As if of type IPSP, speicfy here if its client or server type.

routing-context
: refers to Routing Context already configured on M3UA stack on SGW side. This is optional parameter. If no Routing Context is passed, Application Server Process (assigned to this AS) may not be configured to process signalling traffic related to more than one Application Server, over a single SCTP Association

  Also if ASP is configured to process siganlling traffic from always one AS, irrespective of received messages have routing context set or not, it will always be delieverd to AS for further processing. However if ASP is configured to process siganlling traffic related to more than one AS over a single SCTP Association and signalling message is received without RC, it drops the message and sends back Error message. Respective log4j error is also logged.

traffic-mode
: Traffic mode for ASP's. By default its loadshare. Mobicents M3UA only supports loadshare and override, broadcast is not supported.

Example 4.22. AS (IPSP) creation

```
mobicents(127.0.0.1:3435)>m3ua as create AS1 IPSP mode DE ipspType server rc
 1 traffic-mode loadshare
Successfully created AS name=AS1
```

Example 4.23. AS creation

```
mobicents(127.0.0.1:3435)>m3ua as create AS2 AS mode SE rc 100 traffic-mode
 loadshare
Successfully created AS name=AS2
```

4.4.2.2. Destroy AS

Application Server (AS) can be destroyed by issuing command with following structure:

```
m3ua as destroy <as-name>
```

Where:

as-name
: Simple string name, which identifies AS. Make sure AS is in state INACTIVE and all the ASP's are unassigned before destroying

Example 4.24. Destroy AS

```
mobicents(127.0.0.1:3435)>m3ua as destroy AS1
Successfully destroyed AS name=AS1
```

4.4.2.3. Show AS

Application Server configured can viewed by issuing command with following structure:

```
m3ua as show
```

Example 4.25. Show AS

```
mobicents(127.0.0.1:3435)>m3ua as show
Not supported yet
```

4.4.2.4. Create ASP

Application Server Process (ASP) can be created by issuing command with following structure:

```
m3ua asp create <asp-name> <sctp-association>
```

Where:

asp-name
: Name of this ASP. It should be unique

sctp-association
: name of SCTP Association

Example 4.26. ASP creation

```
mobicents(127.0.0.1:3435)>m3ua asp create ASP1 Assoc1
Successfully created AS name=ASP1
```

4.4.2.5. Destroy ASP

ASP can be destroyed by issuing command with following structure:

```
m3ua asp destroy <asp-name>
```

Where:

asp-name
: Name of this ASP to be destroyed. Make sure ASP is stopped before destroying

Example 4.27. Destroy ASP

```
mobicents(127.0.0.1:3435)>m3ua asp destroy ASP1
Successfully destroyed ASP name=ASP1
```

4.4.2.6. Show ASP

ASP configured can be viewed by issuing command with following structure:

```
m3ua asp show
```

Example 4.28. Show ASP

```
mobicents(127.0.0.1:3435)>m3ua asp show
Not supported yet
```

4.4.2.7. Start ASP

Application Server Process (ASP) can be started with following structure

```
m3ua asp start <asp-name>
```

Where:

asp name
: name of ASP created earlier. Make sure ASP you are trying to start is assigned to at least one AS

Example 4.29. Start ASP

```
mobicents(127.0.0.1:3435)>m3ua asp start ASP1
Successfully started ASP name=ASP1
```

4.4.2.8. Stop ASP

Application Server Process (ASP) can be stopped with following structure

```
m3ua asp stop <asp-name>
```

Where:

asp name
: name of ASP started earlier.

Example 4.30. Stop ASP

```
mobicents(127.0.0.1:3435)>m3ua asp stop ASP1
Successfully stopped ASP name=ASP1
```

4.4.2.9. Add ASP to AS

Application Server Process (ASP) can be assigned to Application Server (AS) with following structure

```
m3ua as add <as-name> <asp-name>
```

Where:

as name
: name of AS created earlier

asp name
: name of ASP created earlier

Note

Mobicents M3UA supports configuring ASP to process signalling traffic related to more than one Application Server, over a single SCTP Association. However you need to make sure that all the AS's that ASP is shared with has Routing Context (unique) configured.

Example 4.31. Add ASP to AS

```
mobicents(127.0.0.1:3435)>m3ua as add AS1 ASP1
Successfully added ASP name=ASP1 to AS name=AS1
```

4.4.2.10. Remove ASP from AS

Application Server Process (ASP) can be unassigned from Application Server (AS) with following structure

```
m3ua as remove <as-name> <asp-name>
```

Where:

as name
: name of AS

asp name
: name of ASP

Example 4.32. Remove ASP from AS

```
mobicents(127.0.0.1:3435)>m3ua as remove AS1 ASP1
Successfully removed ASP name=ASP1 from AS name=AS1
```

4.4.2.11. Add Route

Configure the destination point code that message will be routed to

```
m3ua route add <as-name> <dpc> <opc> <si>
```

Where:

as name
: name of AS created earlier

dpc
: Destination point code

opc
: Originating point code

si
: Service Indicator

Example 4.33. Add Route

```
mobicents(127.0.0.1:3435)>m3ua route add AS1 2 -1 -1
```

4.4.2.12. Remove Route

Remove the As configured for the destination point code

```
m3ua route remove <as-name> <dpc> <opc> <si>
```

Where:

as name
: name of AS assigned to route message for this dpc

dpc
: Destination point code

opc
: Originating point code

si
: Service Indicator

Example 4.34. Remove Route

```
mobicents(127.0.0.1:3435)>m3ua route remove AS1 2 -1 -1
```

4.4.2.13. Show Route

Show all the routes configured

```
m3ua route show
```

Example 4.35. Show Route

```
mobicents(127.0.0.1:3435)>m3ua route show
```

ISUP

ISUP(ISDN User Part or ISUP) is part of SS7 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).

ISUP allows not only to setup a call, but to exchange information about exchange state and its resources(circuits).

Note

Mobicents ISUP is based on ITU-T Q.76X series of documents.

5.1. ISUP Configuration

Mobicents ISUP stack is configured with simple properties. Currently following properties are supported:

Table 5.1. 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

Name	Default value	Value range	Description
t12	15s	15s - 60s	Sets T12 value. Started when BLO is sent. See A.1/Q.764
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

Name	Default value	Value range	Description
			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.

5.2. 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).

5.3. 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;
        }

    }

}
```

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.

6.1. 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.

Routing rules are configured using the Command Line Interface as explained *Section 4.3, "SCCP Management"*

6.1.1. 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 in `sccp rule create` *Section 4.3.1.1, "Create Rule"* command above. Once the digits match actual translation is done

Matching rule

As explained in `sccp rule create` *Section 4.3.1.1, "Create Rule"* command 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

For translation each section in <mask> defined in `sccp rule create` command defines how replacement operation is performed. If <mask> defines K, the originally dialed digits are kept and if <mask> defines R 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 bellow examples

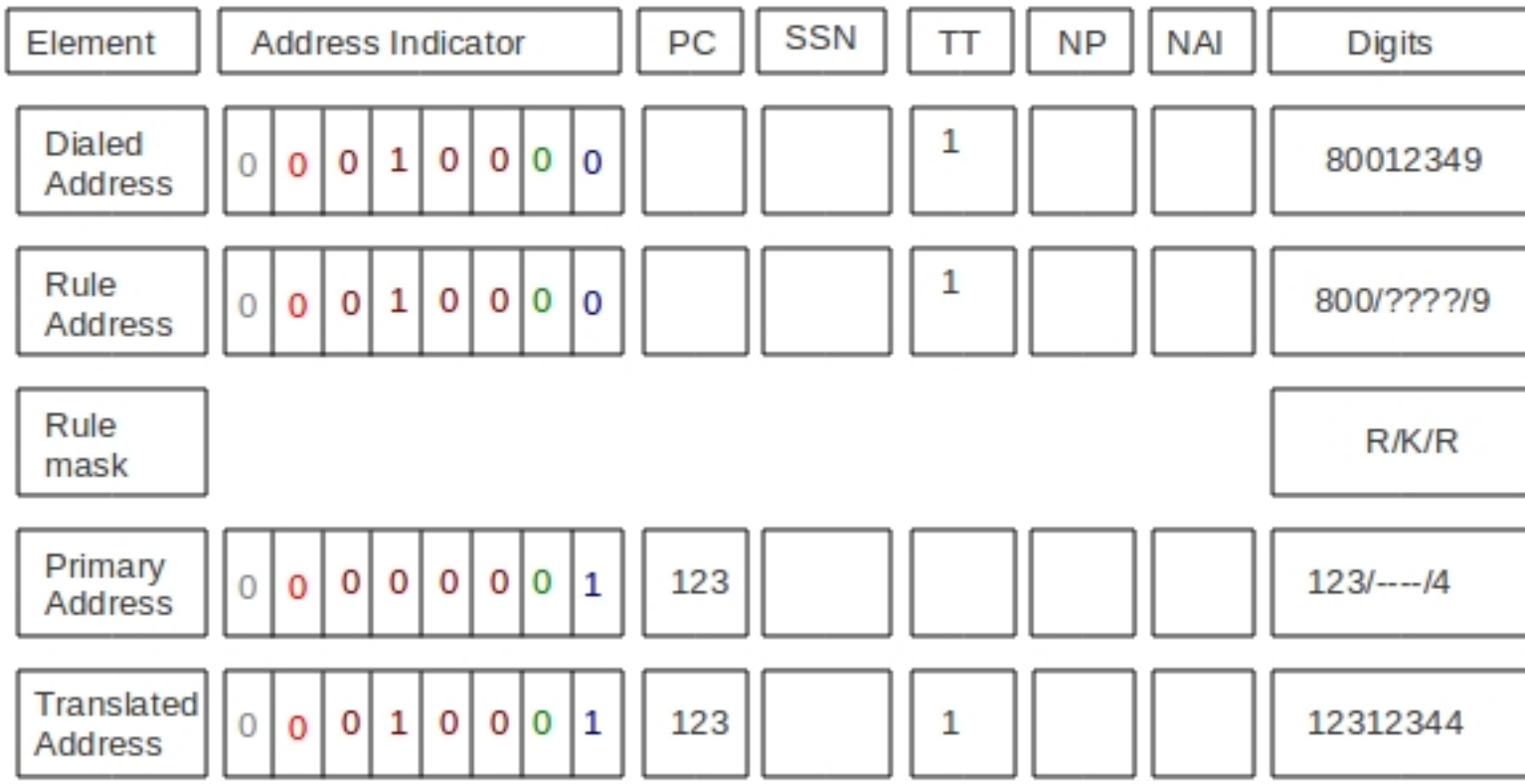
Example 1 : Remove the Global Title and add PC and SSN

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				

GTT - Example 1

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.

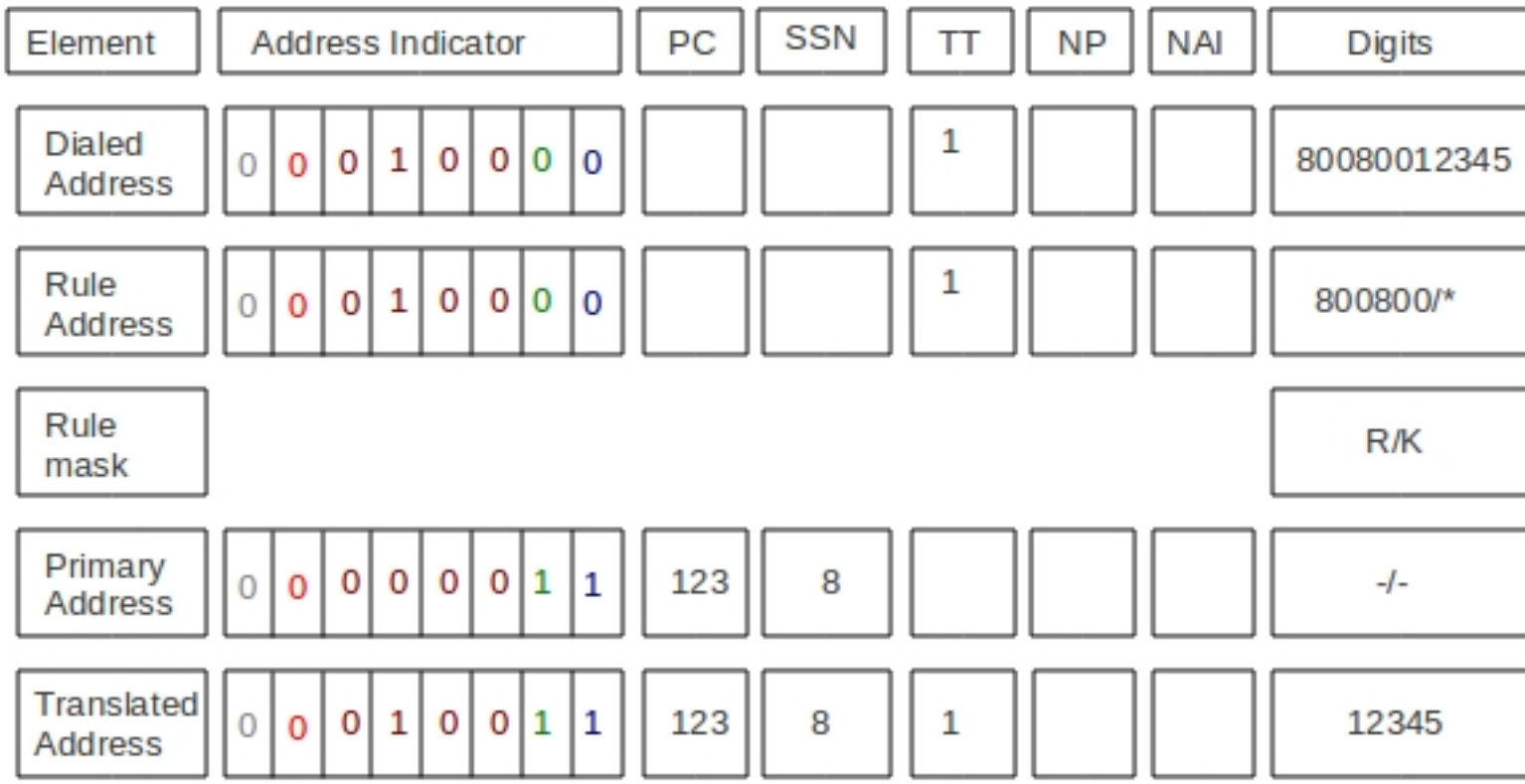


Element	Address Indicator	PC	SSN	TT	NP	NAI	Digits
Dialed Address	0 0 0 1 0 0 0 0			1			80012349
Rule Address	0 0 0 1 0 0 0 0			1			800/????/9
Rule mask							R/K/R
Primary Address	0 0 0 0 0 0 0 1	123					123/----/4
Translated Address	0 0 0 1 0 0 0 1	123		1			12312344

GTT - Example 2

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).

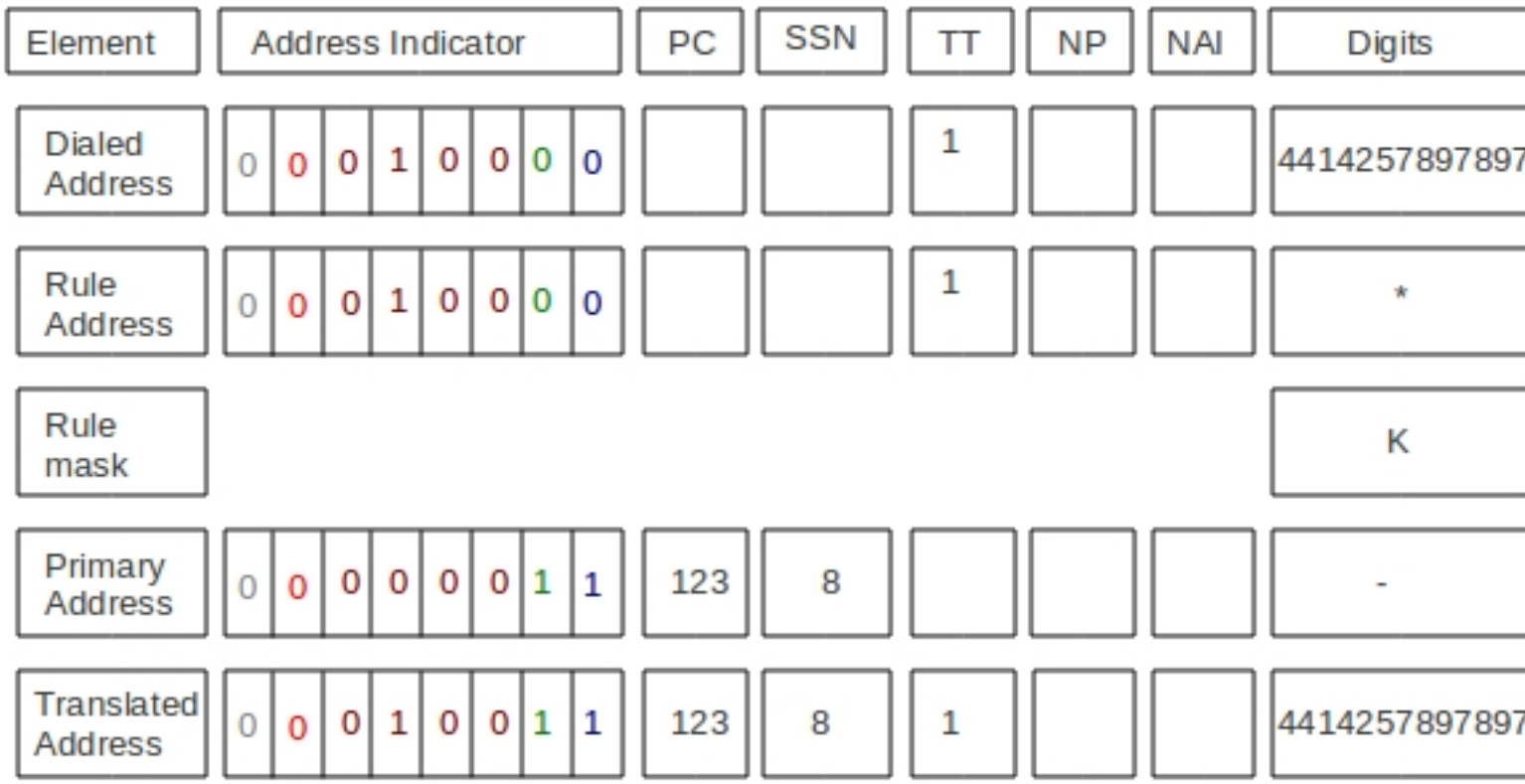


Element	Address Indicator	PC	SSN	TT	NP	NAI	Digits
Dialed Address	0 0 0 1 0 0 0 0			1			80080012345
Rule Address	0 0 0 1 0 0 0 0			1			800800/*
Rule mask							R/K
Primary Address	0 0 0 0 0 0 1 1	123	8				-/-
Translated Address	0 0 0 1 0 0 1 1	123	8	1			12345

GTT - Example 3

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.



Element	Address Indicator	PC	SSN	TT	NP	NAI	Digits
Dialed Address	0 0 0 1 0 0 0 0			1			4414257897897
Rule Address	0 0 0 1 0 0 0 0			1			*
Rule mask							K
Primary Address	0 0 0 0 0 0 1 1	123	8				-
Translated Address	0 0 0 1 0 0 1 1	123	8	1			4414257897897

GTT - Example 4

6.2. 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

Note

The term SCCP User Part refers to the applications that use SCCP's services.

The SCCP User Part gets handle to `SccpStack` by doing JNDI look-up as explained in *Section 6.3, "Access Point"*

`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 `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 concrete SccpMessage viz., `org.mobicents.protocols.ss7.sccp.message.UnitData` or `org.mobicents.protocols.ss7.sccp.message.XUnitData` passing the corresponding parameters created by leveraging ParameterFactory.

6.3. Access Point

`SS7 Service` provides user with access point to `SCCP` protocol/stack.

To get handle to `SccpStack` do the JNDI look-up passing the JNDI name configured in SS7 service as explained in *Section 2.5.5, "Configuring SS7Service"*

```
private static SccpProvider getSccpProvider() throws NamingException {

    // no arg is ok, if we run in JBoss
    InitialContext ctx = new InitialContext();
    try {
        String providerJndiName = "/mobicents/ss7/sccp";
        return ((SccpStack) ctx.lookup(providerJndiName)).getSccpProvider();

    } finally {
        ctx.close();
    }
}
```

6.4. SCCP User Part Example

Below is SCCP User Part example listening for incoming SCCP message and sending back new message

```
public class Test implements SccpListener {

    private SccpProvider sccpProvider;
    private SccpAddress localAddress;
```

```
private static SccpProvider getSccpProvider() throws NamingException {

    // no arg is ok, if we run in JBoss
    InitialContext ctx = new InitialContext();
    try {
        String providerJndiName = "/mobicents/ss7/sccp";
        return ((SccpStack) ctx.lookup(providerJndiName)).getSccpProvider();
    } finally {
        ctx.close();
    }
}

public void start() throws Excetpion {

    this.sccpProvider = getSccpProvider();

    int translationType = 0;
    int subSystemNumber = 0;

    GlobalTitle gt = GlobalTitle.getInstance(translationType,
            NumberingPlan.ISDN_MOBILE, NatureOfAddress.NATIONAL, "1234");

    localAddress = new SccpAddress(gt, 0);

    this.sccpProvider.registerSccpListener(localAddress, this);
}

public void stop() {
    this.sccpProvider.deregisterSccpListener(localAddress);
}

public void onMessage(SccpMessage message) {

    if (message.getType() == MessageType.UDT) {
        throw new IlleagalArgumentException("Dont like UDT");
    } else if (message.getType() == MessageType.XUDT) {
        XUnitData xudt = (XUnitData) message;
        localAddress = ((XUnitData) message).getCalledPartyAddress();
        SccpAddress remoteAddress = ((XUnitData) message)
                .getCallingPartyAddress();

        // now decode content
```

```
            byte[] data = xudt.getData();

            // some data encoded in
            CallRequest cr = new CallRequest(data);

            byte[] answerData;

            if (cr.getCallee().equals(this.localAddress)) {
                EstablihsCallAnswer eca = new EstablihsCallAnswer(cr);
                answerData = eca.encode();

            } else {
                TearDownCallAnswer tdca = new TearDownCallAnswer(cr);
                answerData = tdca.encode();
            }

            HopCounter hc = this.sccpProvider.getParameterFactory()
                    .createHopCounter(5);

            XUnitData sccpAnswer = this.sccpProvider
                    .getMessageFactory()
                    .createXUnitData(hc, xudt.getProtocolClass(),
                            message.getCallingPartyAddress(), this.localAddress);

            this.sccpProvider.send(sccpAnswer);

        }

    }

}
```

TCAP

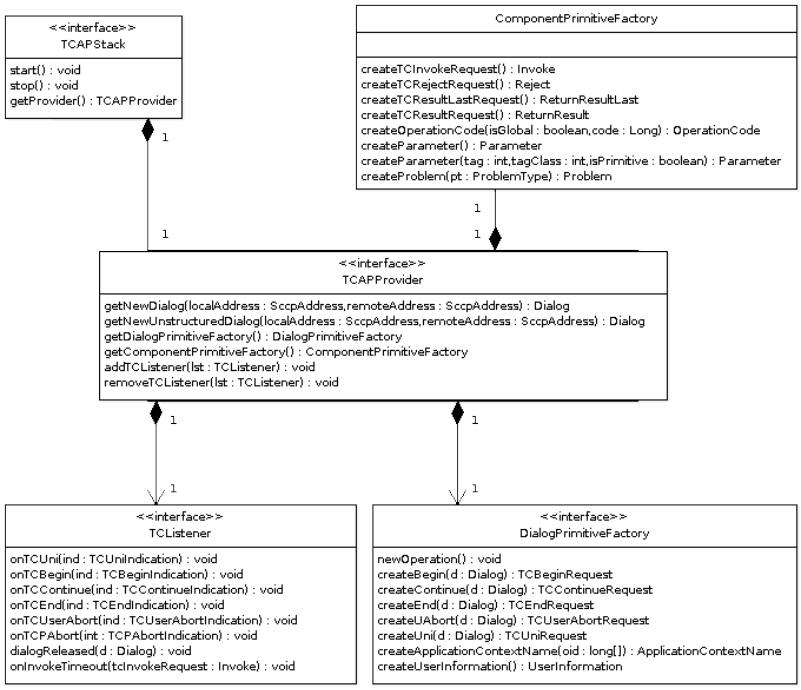
The Transaction Capabilities Application Part (TCAP) is defined in ITU-T Recommendations Q.771-Q.775. TCAP allows services at network nodes to communicate with each other using an agreed-upon set of data elements. Its primary purpose is to facilitate multiple concurrent dialogs between the same sub-systems 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.

7.1. Mobicents SS7 Stack TCAP Usage

The `org.mobicents.protocols.ss7.tcap.api.TCAPStack` interface defines the methods required to represent TCAP Protocol Stack. TCAPStack exposes `org.mobicents.protocols.ss7.tcap.api.TCAPProvider` that interacts directly with 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 network. TCAP User Part also allows to register`org.mobicents.protocols.ss7.tcap.api.TCListener` to listen 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 looks like



Mobicents SS7 Stack TCAP Class Diagram

The `org.mobicents.protocols.ss7.tcap.TCAPStackImpl` is concrete implementation of `TCAPStack`. The TCAP User Part creates instance of `TCAPStackImpl` passing the reference of `SccpProvider` and new instance of `SccpAddress` representing address to which bind listener. The `TCAP` stack creates internaly Mobicents `MAP` Stack implementation. Passed `SccpAddress` is used to match against incoming messages destination address.

```
        SccpProvider sccpProvider = getSccpProvider(); //JNDI lookup of SCCP Stack and get
Provider
        SccpAddress localAddress createLocalAddress();

        TCAPStack tcapStack = new TCAPStackImpl(sccpPprovider, localAddress);


        ...

        private SccpAddress createLocalAddress()
        {

           return new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 1, null, 8);
        }
```

The reference to `SccpProvider` is received from `SccpStack`. To get handle to `SccpStack` do the JNDI look-up passing the JNDI name configured in SS7 service as explained in *Section 6.3, "Access Point"*

The TCAP User Part should register the concrete implementation of `TCListener` with `TCAPProvider` to listen for incoming TCAP messages.

```
  public class Client implements TCListener{
        .....
        tcapProvider = tcapStack.getProvider();
        tcapProvider.addTCListener(this);
        ....
  }
```

The TCAP User Part leverages TCAPProvider to create new Dialog. The component's between the nodes are exchanged within this Dialog

```
clientDialog = this.tcapProvider.getNewDialog(thisAddress, remoteAddress);
```

The TCAP User Part leverages ComponentPrimitiveFactory to create new components. These components are sent usig the dialog

```
//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);
```

7.2. Mobicents SS7 Stack TCAP User Part Example

Below is TCAP User Part example. This example creates dialog and exchanges messages withing structured dialog. Refer to source for function calls:

```
public class Client implements TCListener{
  //encoded Application Context Name
  public static final long[] _ACN_ = new long[] { 0, 4, 0, 0, 1, 0, 19, 2 };

  private TCAPStack stack;
  private SccpAddress thisAddress;
  private SccpAddress remoteAddress;

  private TCAPProvider tcapProvider;
  private Dialog clientDialog;
```

```
  Client(SccpProvider sccpPprovider, SccpAddress thisAddress,SccpAddress remoteAddress) {
      super();
      this.stack = new TCAPStackImpl(sccpPprovider,thisAddress); //pass address, so stack can
register in SCCP
      this.runningTestCase = runningTestCase;
      this.thisAddress = thisAddress;
      this.remoteAddress = remoteAddress;
      this.tcapProvider = this.stack.getProvider();
      this.tcapProvider.addTCListener(this);
  }

  private static SccpProvider getSccpProvider() throws NamingException {

      // no arg is ok, if we run in JBoss
      InitialContext ctx = new InitialContext();
      try {
          String providerJndiName = "/mobicents/ss7/sccp";
          return ((SccpStack) ctx.lookup(providerJndiName)).getSccpProvider();
      } finally {
          ctx.close();
      }
  }

  public void start() throws TCAPException, TCAPSendException {
      clientDialog = this.tcapProvider.getNewDialog(thisAddress, 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)
{
    d.keepAlive();
}


public void onInvokeTimeout(Invoke tcInvokeRequest)
{
}


public void onDialogTimeout(Dialog d)
{
}

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 static void main(String[] args)
    {

        SccpAddress localAddress = new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 1


        SccpAddress remoteAddress = new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN,
        Client c = new Client(getSccpProvider(),localAddress,remoteAddress);
    }

}
```

MAP

Mobile application part (MAP) is the protocol that is used to allow the GSM network nodes within the Network Switching Subsystem (NSS) to communicate with each other to provide services, such as roaming capability, text messaging (SMS), Unstructured Supplementary Service Data (USSD) and subscriber authentication. MAP provides an application layer on which to build the services that support a GSM network. This application layer provides a standardized set of services. MAP uses the services of the SS7 network, specifically the Signaling Connection Control Part (SCCP) and the Transaction Capabilities Application Part (TCAP)

Important

For better understanding of this chapter please read GSM 09.02.

Note

Mobicents SS7 Stack MAP has implementation for USSD, SMS and Location Management Service (LMS) Messages only. Any contribution to implement other messages are welcome. We will provide you all the help that you may need initially.

8.1. SS7 Stack MAP

The `org.mobicents.protocols.ss7.map.api.MAPStack` interface defines the methods required to represent MAP Protocol Stack. MAPStack exposes `org.mobicents.protocols.ss7.map.api.MAPProvider` that interacts directly with MAPStack. This interface defines the methods that will be used by any registered MAP User application implementing the `org.mobicents.protocols.ss7.map.api.MAPDialogListener` and `org.mobicents.protocols.ss7.map.api.MAPServiceListener` interface to listen MAP messages and dialogue handling primitives.

Each MAP-User interested in listening messages specific to MAP Service implements specific `MAPServiceListener`.

- MAP-User interested only in USSD messages implements `org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListener`
- MAP-User interested only in SMS messages implements `org.mobicents.protocols.ss7.map.api.service.sms.MAPServiceSmsListener`
- MAP-User interested only in USSD messages implements `org.mobicents.protocols.ss7.map.api.service.lsm.MAPServiceLsmListener`

MAP-User interested in all the services may implement all the service listener class.

The `org.mobicents.protocols.ss7.map.MAPStackImpl` is concrete implementation of `MAPStack`. The MAP User application creates instance of `MAPStackImpl` passing the reference of `SccpProvider` and Sub System Number. All incoming messages are checked for destination SSN, if it matches with the one registered with this `MAPStackImpl` the corresponding listener is called else the peer receives error.

```
SccpProvider sccpProvider = getSccpProvider(); //JNDI lookup of SCCP Stack and get Provider
    MAPStackImpl mapStack = new MAPStackImpl(sccpPprovider, 8);
    ...
```

The reference to `SccpProvider` is received from `SccpStack`. To get handle to `SccpStack` do the JNDI look-up passing the JNDI name configured in SS7 service as explained in *Section 6.3, "Access Point"*

The MAP User application should register the concrete implementation of `MAPDialogListener` with `MAPProvider` to listen for incoming MAP Dialog and MAP Primitive messages.

The MAP User application should register the concrete implementation of `MAPServiceListener` with corresponding `MAPServiceBase` to listen for incoming MAP Service messages. Following `MAPServiceBase` are exposed by `MAPProvider`

- For LSM service `org.mobicents.protocols.ss7.map.api.service.lsm.MAPServiceLsm`
- For SMS service `org.mobicents.protocols.ss7.map.api.service.sms.MAPServiceSms`
- For USSD service `org.mobicents.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementary`

```
public class MAPExample implements MAPDialogListener, MAPServiceSupplementaryListener {
    .....
    mapProvider = mapStack.getMAPProvider();
    mapProvider.addMAPDialogListener(this);
    mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
    ....
}
```

Before any MAP specific service can be used, the corresponding service should be activated

```
    .....
```

```
// Make the supplimentary service activated
mapProvider.getMAPServiceSupplementary().acivate();
....
```

The MAP User Application leverages `MapServiceFactory` to create instance of `USSDString` and `AddressString`

```
MapServiceFactory servFact = mapProvider.getMapServiceFactory();
USSDString ussdString = servFact.createUSSDString("*125*+31628839999#",
    null);

AddressString msisdn = this.servFact.createAddressString(
    AddressNature.international_number, NumberingPlan.ISDN,
    "31628838002");
```

The MAP User Application leverages specific `MAPServiceBase` to create new `MAPDialog` and send message

```
// First create Dialog
MAPDialogSupplementary mapDialog = mapProvider.getMAPServiceSupplementary().createNewDialog(
    MAPApplicationContext.getInstance(MAPApplicationContextName.networkUnstructuredSsContext, MAPA
    destReference, origAddress, origReference);

// The dataCodingScheme is still byte, as I am not exactly getting how
// to encode/decode this.
byte ussdDataCodingScheme = 0x0f;

// USSD String: *125*+31628839999#
// 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 = servFact.createUSSDString("*125*+31628839999#", null);

AddressString msisdn = this.servFact.createAddressString(AddressNature.international_number,
    NumberingPlan.ISDN, "31628838002");

mapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme, ussdString, msisdn);
```

```
        // This will initiate the TC-BEGIN with INVOKE component
        mapDialog.send();
```

8.2. SS7 Stack MAP Usage

The complete example looks like

```
public class MAPExample implements MAPDialogListener, MAPServiceSupplementaryListener {

    private MAPStack mapStack;
    private MAPProvider mapProvider;

    MapServiceFactory servFact;

    SccpAddress destAddress = null;

    // The address created by passing the AddressNature, NumberingPlan and
    // actual address
    AddressString destReference = servFact.createAddressString(AddressNature.international_number,
            NumberingPlan.land_mobile, "204208300008002");

    SccpAddress origAddress = null;

    AddressString origReference = servFact.createAddressString(AddressNature.international_number, NumberingP
            "31628968300");

    MAPExample(SccpProvider sccpPprovider, SccpAddress address, SccpAddress remoteAddress) {
        origAddress = address;
        destAddress = remoteAddress;

        mapStack = new MAPStackImpl(sccpPprovider, 8);
        mapProvider = mapStack.getMAPProvider();
        servFact = mapProvider.getMapServiceFactory();

        mapProvider.addMAPDialogListener(this);
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
    }

    private static SccpProvider getSccpProvider() throws NamingException {

        // no arg is ok, if we run in JBoss
```

```
        InitialContext ctx = new InitialContext();
        try {
            String providerJndiName = "/mobicents/ss7/sccp";
            return ((SccpStack) ctx.lookup(providerJndiName)).getSccpProvider();

        } finally {
            ctx.close();
        }
    }

    private static SccpAddress createLocalAddress() {
        return new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 1, null, 8);
    }

    private static SccpAddress createRemoteAddress() {
        return new SccpAddress(RoutingIndicator.ROUTING_BASED_ON_DPC_AND_SSN, 2, null, 8);
    }

    public void run() throws Exception {

        // Make the supplimentary service activated
        mapProvider.getMAPServiceSupplementary().acivate();

        // First create Dialog
        MAPDialogSupplementary mapDialog = mapProvider.getMAPServiceSupplementary().createNewDialog(
                MAPApplicationContext.getInstance(MAPApplicationContextName.networkUnstructuredSsContext, MAPA
                destReference, origAddress, origReference);

        // The dataCodingScheme is still byte, as I am not exactly getting how
        // to encode/decode this.
        byte ussdDataCodingScheme = 0x0f;

        // USSD String: *125*+31628839999#
        // 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 = servFact.createUSSDString("*125*+31628839999#", null);

        AddressString msisdn = this.servFact.createAddressString(AddressNature.international_number,
                NumberingPlan.ISDN, "31628838002");

        mapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme, ussdString, msisdn);

        // This will initiate the TC-BEGIN with INVOKE component
```

```
        mapDialog.send();
    }

    public void onProcessUnstructuredSSIndication(ProcessUnstructuredSSIndication procUnstrInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSIndication(UnstructuredSSIndication unstrInd) {
        // TODO Auto-generated method stub

    }

    public static void main(String[] args) throws Exception {
        SccpProvider sccpProvider = getSccpProvider(); // JNDI lookup of SCCP

        SccpAddress localAddress = createLocalAddress();
        SccpAddress remoteAddress = createRemoteAddress();

        MAPExample example = new MAPExample(sccpProvider, localAddress, remoteAddress);

        example.run();

    }

    @Override
    public void onDialogRequest(MAPDialog mapDialog, AddressString destReference, AddressString origReferenc
            MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogAccept(MAPDialog mapDialog, MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogReject(MAPDialog mapDialog, MAPRefuseReason refuseReason, MAPProviderError provid
            ApplicationContextName alternativeApplicationContext, MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }
```

```
@Override
public void onDialogUserAbort(MAPDialog mapDialog, MAPUserAbortChoice userReason,
        MAPExtensionContainer extensionContainer) {
    // TODO Auto-generated method stub

}

@Override
public void onDialogProviderAbort(MAPDialog mapDialog, MAPAbortProviderReason abortProviderReason,
        MAPAbortSource abortSource, MAPExtensionContainer extensionContainer) {
    // TODO Auto-generated method stub

}

@Override
public void onDialogClose(MAPDialog mapDialog) {
    // TODO Auto-generated method stub

}

@Override
public void onDialogDelimiter(MAPDialog mapDialog) {
    // TODO Auto-generated method stub

}

@Override
public void onDialogNotice(MAPDialog mapDialog, MAPNoticeProblemDiagnostic noticeProblemDiagnostic) {
    // TODO Auto-generated method stub

}

public void onDialogResease(MAPDialog mapDialog) {

}

@Override
public void onDialogTimeout(MAPDialog mapDialog) {
    // TODO Auto-generated method stub

}

@Override
```

```
    public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage mapErrorMessage) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onProviderErrorComponent(MAPDialog mapDialog, Long invokeId, MAPProviderError providerError)
        // TODO Auto-generated method stub

    }

    @Override
    public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onInvokeTimeout(MAPDialog mapDialog, Long invoke) {
        // TODO Auto-generated method stub

    }

}
```

Appendix A. Java Development Kit (JDK): Installing, Configuring and Running

The **Mobicents Platform** is written in Java; therefore, before running any **Mobicents** server, you must have a working Java Runtime Environment (JRE) or Java Development Kit (JDK) installed on your system. In addition, the JRE or JDK you are using to run **Mobicents** must be version 5 or higher[1].

Should I Install the JRE or JDK? Although you can run **Mobicents** servers using the Java Runtime Environment, we assume that most users are developers interested in developing Java-based, **Mobicents**-driven solutions. Therefore, in this guide we take the tact of showing how to install the full Java Development Kit.

Should I Install the 32-Bit or the 64-Bit JDK, and Does It Matter? Briefly stated: if you are running on a 64-Bit Linux or Windows platform, you should consider installing and running the 64-bit JDK over the 32-bit one. Here are some heuristics for determining whether you would rather run the 64-bit Java Virtual Machine (JVM) over its 32-bit cousin for your application:

- Wider datapath: the pipe between RAM and CPU is doubled, which improves the performance of memory-bound applications when using a 64-bit JVM.
- 64-bit memory addressing gives virtually unlimited (1 exabyte) heap allocation. However large heaps affect garbage collection.
- Applications that run with more than 1.5 GB of RAM (including free space for garbage collection optimization) should utilize the 64-bit JVM.
- Applications that run on a 32-bit JVM and do not require more than minimal heap sizes will gain nothing from a 64-bit JVM. Barring memory issues, 64-bit hardware with the same relative clock speed and architecture is not likely to run Java applications faster than their 32-bit cousin.

Note that the following instructions detail how to download and install the 32-bit JDK, although the steps are nearly identical for installing the 64-bit version.

Downloading. You can download the Sun JDK 5.0 (Java 2 Development Kit) from Sun's website: *http://java.sun.com/javase/downloads/index_jdk5.jsp*. Click on the **Download** link next to "JDK 5.0 Update <*x*>" (where <*x*> is the latest minor version release number). On the next page, select your language and platform (both architecture—whether 32- or 64-bit—and operating

[1] At this point in time, it is possible to run most **Mobicents** servers, such as the JAIN SLEE, using a Java 6 JRE or JDK. Be aware, however, that presently the XML Document Management Server does not run on Java 6. We suggest checking the Mobicents web site, forums or discussion pages if you need to inquire about the status of running the XML Document Management Server with Java 6.

system), read and agree to the `Java Development Kit 5.0 License Agreement`, and proceed to the download page.

The Sun website will present two download alternatives to you: one is an RPM inside a self-extracting file (for example, `jdk-1_5_0_16-linux-i586-rpm.bin`), and the other is merely a self-extracting file (e.g. `jdk-1_5_0_16-linux-i586.bin`). If you are installing the JDK on Red Hat Enterprise Linux, Fedora, or another RPM-based Linux system, we suggest that you download the self-extracting file containing the RPM package, which will set up and use the SysV service scripts in addition to installing the JDK. We also suggest installing the self-extracting RPM file if you will be running **Mobicents** in a production environment.

Installing. The following procedures detail how to install the Java Development Kit on both Linux and Windows.

Procedure A.1. Installing the JDK on Linux

- Regardless of which file you downloaded, you can install it on Linux by simply making sure the file is executable and then running it:

```
~]$ chmod +x "jdk-1_5_0_<minor_version>-linux-<architecture>-rpm.bin"
~]$ ./"jdk-1_5_0_<minor_version>-linux-<architecture>-rpm.bin"
```

You Installed Using the Non-RPM Installer, but Want the SysV Service Scripts

If you download the non-RPM self-extracting file (and installed it), and you are running on an RPM-based system, you can still set up the SysV service scripts by downloading and installing one of the `-compat` packages from the JPackage project. Remember to download the `-compat` package which corresponds correctly to the minor release number of the JDK you installed. The compat packages are available from *ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/*.

Important

You do not need to install a `-compat` package in addition to the JDK if you installed the self-extracting RPM file! The `-compat` package merely performs the same SysV service script set up that the RPM version of the JDK installer does.

Procedure A.2. Installing the JDK on Windows

- Using Explorer, simply double-click the downloaded self-extracting installer and follow the instructions to install the JDK.

Configuring. Configuring your system for the JDK consists in two tasks: setting the `JAVA_HOME` environment variable, and ensuring that the system is using the proper JDK (or JRE) using the `alternatives` command. Setting `JAVA_HOME` usually overrides the values for `java`, `javac` and `java_sdk_1.5.0` in `alternatives`, but we will set them all just to be safe and consistent.

Setting the `JAVA_HOME` Environment Variable on Generic Linux

After installing the JDK, you must ensure that the `JAVA_HOME` environment variable exists and points to the location of your JDK installation.

Setting the `JAVA_HOME` Environment Variable on Linux. You can determine whether `JAVA_HOME` is set on your system by `echo`ing it on the command line:

```
~]$ echo $JAVA_HOME
```

If `JAVA_HOME` is not set already, then you must set its value to the location of the JDK installation on your system. You can do this by adding two lines to your personal `~/.bashrc` configuration file. Open `~/.bashrc` (or create it if it doesn't exist) and add a line similar to the following one anywhere inside the file:

```
export JAVA_HOME="/usr/lib/jvm/jdk1.5.0_<version>"
```

You should also set this environment variable for any other users who will be running **Mobicents** (any environment variables `exported` from `~/.bashrc` files are local to that user).

Setting `java`, `javac` and `java_sdk_1.5.0` Using the `alternatives` command

Selecting the Correct System JVM on Linux using `alternatives` . On systems with the `alternatives` command, including Red Hat Enterprise Linux and Fedora, you can easily choose which JDK (or JRE) installation you wish to use, as well as which `java` and `javac` executables should be run when called.

As the root user, call `/usr/sbin/alternatives` with the `--config java` option to select between JDKs and JREs installed on your system:

```
root@localhost ~]$ /usr/sbin/alternatives --config java

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           /usr/lib/jvm/jre-1.5.0-gcj/bin/java
   2           /usr/lib/jvm/jre-1.6.0-sun/bin/java
*+ 3           /usr/lib/jvm/jre-1.5.0-sun/bin/java
```

```
Enter to keep the current selection[+], or type selection number:
```

In our case, we want to use the Sun JDK, version 5, that we downloaded and installed, to run the `java` executable. In the `alternatives` information printout above, a plus (+) next to a number indicates the one currently being used. As per `alternatives`' instructions, pressing **Enter** will simply keep the current JVM, or you can enter the number corresponding to the JVM you would prefer to use.

Repeat the procedure above for the `javac` command and the `java_sdk_1.5.0` environment variable, *as the root user*:

```
~]$ /usr/sbin/alternatives --config javac
```

```
~]$ /usr/sbin/alternatives --config java_sdk_1.5.0
```

Setting the `JAVA_HOME` Environment Variable on Windows

For information on how to set environment variables in Windows, refer to *http://support.microsoft.com/kb/931715*.

Testing. Finally, to make sure that you are using the correct JDK or Java version (5 or higher), and that the java executable is in your `PATH`, run the `java -version` command in the terminal from your home directory:

```
~]$ java -version
java version "1.5.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_16-b03)
Java HotSpot(TM) Client VM (build 1.5.0_16-b03, mixed mode, sharing)
```

Uninstalling. There is usually no reason (other than space concerns) to remove a particular JDK from your system, given that you can switch between JDKs and JREs easily using `alternatives`, and/or by setting `JAVA_HOME`.

Uninstalling the JDK on Linux. On RPM-based systems, you can uninstall the JDK using the `yum remove <jdk_rpm_name>` command.

Uninstalling the JDK on Windows. On Windows systems, check the JDK entry in the `Start` menu for an uninstall command, or use `Add/Remove Programs`.

Appendix B. Setting the JBOSS_HOME Environment Variable

The **Mobicents Platform** (**Mobicents**) is built on top of the **JBoss Application Server**. You do not need to set the `JBOSS_HOME` environment variable to run any of the **Mobicents Platform** servers *unless* `JBOSS_HOME` is *already* set.

The best way to know for sure whether `JBOSS_HOME` was set previously or not is to perform a simple check which may save you time and frustration.

Checking to See If JBOSS_HOME is Set on Unix. At the command line, `echo` **`$JBOSS_HOME`** to see if it is currently defined in your environment:

```
~]$ echo $JBOSS_HOME
```

The **Mobicents Platform** and most Mobicents servers are built on top of the **JBoss Application Server** (**JBoss Application Server**). When the **Mobicents Platform** or Mobicents servers are built *from source*, then `JBOSS_HOME` *must* be set, because the Mobicents files are installed into (or "over top of" if you prefer) a clean **JBoss Application Server** installation, and the build process assumes that the location pointed to by the `JBOSS_HOME` environment variable at the time of building is the **JBoss Application Server** installation into which you want it to install the Mobicents files.

This guide does not detail building the **Mobicents Platform** or any Mobicents servers from source. It is nevertheless useful to understand the role played by **JBoss AS** and `JBOSS_HOME` in the Mobicents ecosystem.

The immediately-following section considers whether you need to set `JBOSS_HOME` at all and, if so, when. The subsequent sections detail how to set `JBOSS_HOME` on Unix and Windows

Important

Even if you fall into the category below of *not needing* to set `JBOSS_HOME`, you may want to for various reasons anyway. Also, even if you are instructed that you do *not need* to set `JBOSS_HOME`, it is good practice nonetheless to check and make sure that `JBOSS_HOME` actually *isn't* set or defined on your system for some reason. This can save you both time and frustration.

You *DO NOT NEED* to set `JBOSS_HOME` if...

- ...you have installed the **Mobicents Platform** binary distribution.

- ...you have installed a Mobicents server binary distribution *which bundles* ***JBoss Application Server***.

You *MUST* set `JBOSS_HOME` if...

- ...you are installing the **Mobicents Platform** or any of the Mobicents servers *from source*.
- ...you are installing the **Mobicents Platform** binary distribution, or one of the Mobicents server binary distributions, which *do not* bundle **JBoss Application Server**.

Naturally, if you installed the **Mobicents Platform** or one of the Mobicents server binary releases which *do not* bundle **JBoss Application Server**, yet requires it to run, then you should install before setting `JBOSS_HOME` or proceeding with anything else.

Setting the JBOSS_HOME Environment Variable on Unix. The `JBOSS_HOME` environment variable must point to the directory which contains all of the files for the **Mobicents Platform** or individual Mobicents server that you installed. As another hint, this topmost directory contains a `bin` subdirectory.

Setting `JBOSS_HOME` in your personal `~/.bashrc` startup script carries the advantage of retaining effect over reboots. Each time you log in, the environment variable is sure to be set for you, as a user. On Unix, it is possible to set `JBOSS_HOME` as a system-wide environment variable, by defining it in `/etc/bashrc`, but this method is neither recommended nor detailed in these instructions.

Procedure B.1. To Set JBOSS_HOME on Unix...

1. Open the `~/.bashrc` startup script, which is a hidden file in your home directory, in a text editor, and insert the following line on its own line while substituting for the actual install location on your system:

```
export JBOSS_HOME="/home/<username>/<path>/<to>/<install_directory>"
```

2. Save and close the `.bashrc` startup script.
3. You should `source` the `.bashrc` script to force your change to take effect, so that `JBOSS_HOME` becomes set for the current session[1].

```
~]$ source ~/.bashrc
```

4. Finally, ensure that `JBOSS_HOME` is set in the current session, and actually points to the correct location:

[1] Note that any other terminals which were opened prior to your having altered `.bashrc` will need to `source ~/.bashrc` as well should they require access to `JBOSS_HOME`.

Note

The command line usage below is based upon a binary installation of the **Mobicents Platform**. In this sample output, JBOSS_HOME has been set correctly to the *topmost_directory* of the **Mobicents** installation. Note that if you are installing one of the standalone **Mobicents** servers (with **JBoss AS** bundled!), then JBOSS_HOME would point to the *topmost_directory* of your server installation.

```
~]$ echo $JBOSS_HOME
/home/silas/<path>/<to>/<install_directory>
```

Setting the JBOSS_HOME Environment Variable on Windows. The JBOSS_HOME environment variable must point to the directory which contains all of the files for the Mobicents Platform or individual Mobicents server that you installed. As another hint, this topmost directory contains a bin subdirectory.

For information on how to set environment variables in recent versions of Windows, refer to *http://support.microsoft.com/kb/931715*.

Appendix C. Revision History

Revision History		
Revision 1.0	Wed June 2 2010	BartoszBaranowski
Creation of the Mobicents SS7 Stack User Guide.		
Revision 1.1	Tue Dec 21 2010	AmitBhayani
Creation of the Mobicents SS7 Stack User Guide.		

Index

F

feedback, viii