Mobicents SS7 Stack User Guide

by Amit Bhayani, Bartosz Baranowski, and Oleg Kulikov

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/] .
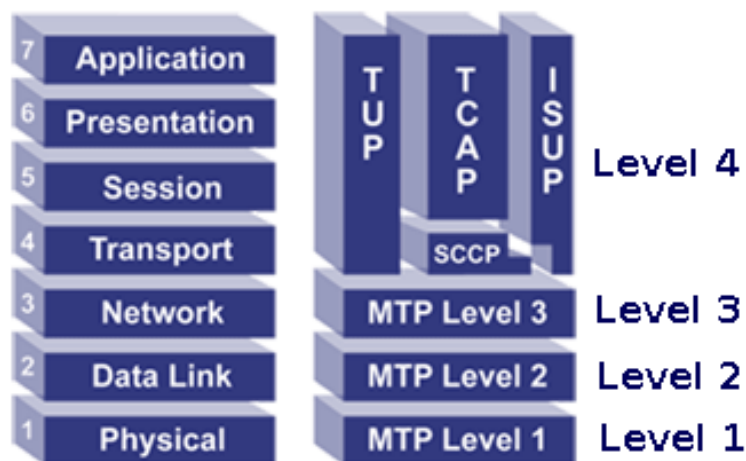
SS7 Stack overview

Mobicents SS7 Stack is software based SS7 protocol implementation providing Level 2 and above.

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.

The Basics

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 any SS7 hardware available in the market and yet Mobicents SS7 Stack Level 4 and above remains the same.

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

Mobicents SS7 Stack consists of following functional blocks:

2.1. Linkset

`Linkset` is logical group of links between two Signaling Points. This term hides `MTP` layer, ie. it abstracts whether hardware `MTP` is used , M3UA or Mobicents SS7 Stack one.

Three type of `Linkset` are defined

- `DahdiLinkset` : As the name suggests, this linkset is for `dahdi` based hardware. `dahdi` boards only provides SS7 MTP1 layer and usually depends on external software to provide MTP2/MTP3 support.

  Well known `dahdi` based SS7 cards are `Diguim` and `Sangoma`

- `DialogicLinkset` : Linkset for `dialogic` based hardware. `dialogic` boards have MTP2 and MTP3 support on board.

- `M3UALinkset` : M3UA stands for MTP Level 3 (MTP3) User Adaptation Layer as defined by the IETF SIGTRAN working group in RFC 4666. `M3UALinkset` enables the Mobicents SS7 Stack Level 4 (e.g. ISUP, SCCP) to run over IP instead of instead of SS7 network

Each type of linkset has coresponding factory. Please refer to *Section 3.2.1, "Configuring LinksetFactory"* for list and guidelines how to configure each factory.

Configuration of linkset factories is exaplained in section *Section 3.2, " Configuring Mobicents SS7 Stack "*

2.2. Shell Management client

`Shell` is Command Line Interface (CLI) tool which allows to manage different aspects of Mobicents SS7 Stack in interactive manner. It connects to different instances of Mobicents SS7 Stack which

manage `Linksets` and `SCCP` routing. For detailed information please refer to: *???*. Usually `Shell` will be invoked from remote machine(remote to `Linksets` and application protocols).

2.3. SS7 Service

Service is element which is used to manage `LinkSets` and protocols like `SCCP`. SS7 service creates instance of Mobicents SS7 Stack SCCP and bind's it to JNDI name java:/mobicents/ss7/sccp

SS7 Service is JMX based service deployed in JBoss Application Server

Diagram below depicts elements which are deployed as part of `SS7 Service`:
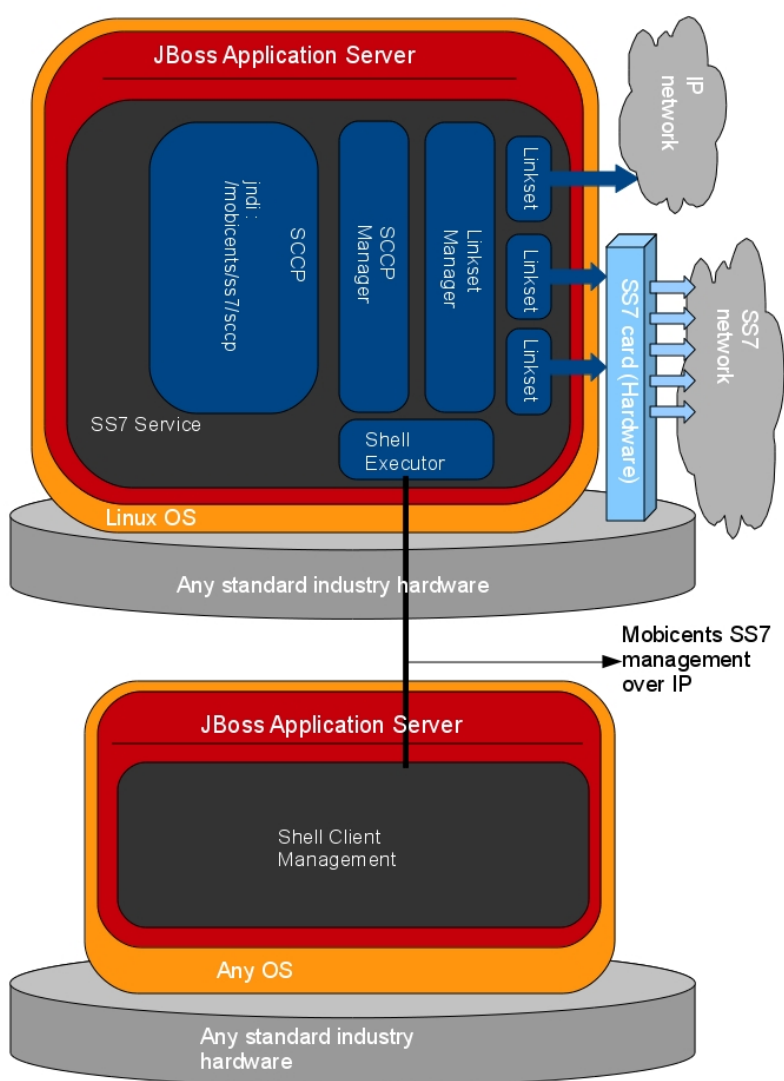
Mobicents SS7 Stack SS7Service elements.

Service serves following purposes:

Expose protocol access points

Access points allows user to access lower layer protocols, like `SCCP` and interact through such protocols with `SS7` network.

Expose management interface

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

Manage Linksets and their lifecycle

`Linkset Manager` persists linksets information and manages their lifecycle, ie. it creates, activates them(also after restart).

Configuration of SS7 Service is exaplained in section *Section 3.2, " Configuring Mobicents SS7 Stack "*

2.4. Mobicents SS7 Stack Usage

Diagram below depicts how Mobicents SS7 Stack is used:
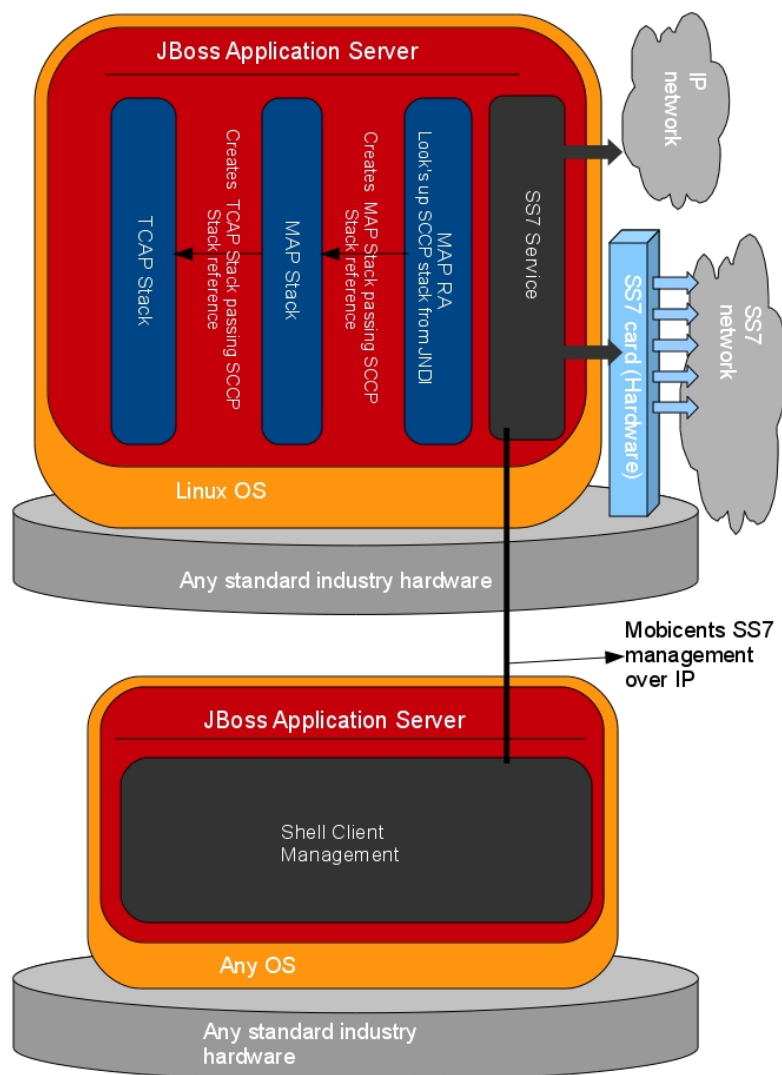
Mobicents SS7 Stack general design

Installation and Running

3.1. Installing

Mobicents SS7 stack at its core requires only Java 1.5(Java SE) if you are using only `M3UALinkset`. However if you plan to use `DahdiLinkset` or `DialogicLinkset`, 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

3.1.1. Binary Service

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

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

- `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.

- `ss7` : ss7 contains the service that is deployed in JBoss AS and libraries that end applications refers to. ss7 is further divided into sub-modules

  - `docs` : User guide for Mobicents SS7 Stack

  - `mobicents-ss7-service` : SS7 service is the core engine as explained in section *Section 2.3, "SS7 Service"*

  - `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 `M3UALinkset` 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 Stack. Refer *???* to understand how to use shell

Binary release has following layout:

```
▽ 📁 mobicents-ss7
    ▽ 📁 asn
        ▷ 📁 docs
            📦 asn-1.0.0-SNAPSHOT.jar
    ▽ 📁 ss7
        ▷ 📁 docs
        ▽ 📁 mobicents-ss7-service
            ▷ 📁 lib
            ▽ 📁 META-INF
                📄 jboss-beans.xml
                📄 jboss-structure.xml
            📄 sccp-routing.txt
        ▽ 📁 native
            📄 libmobicents-dahdi-linux.so
        ▽ 📁 protocols
            📦 isup-api-1.0.0-SNAPSHOT.jar
            📦 isup-impl-1.0.0-SNAPSHOT.jar
            📦 map-api-1.0.0-SNAPSHOT.jar
            📦 map-impl-1.0.0-SNAPSHOT.jar
            📦 tcap-api-1.0.0-SNAPSHOT.jar
            📦 tcap-impl-1.0.0-SNAPSHOT.jar
        ▽ 📁 shell
            📦 mobicents-ss7-shell.jar
            📄 ss7-run.bat
            📄 ss7-run.sh
        📄 build.xml
    📄 README.TXT
```

Mobicents SS7 Stack binary layout.

3.1.2. Installing Binary

The Mobicents SS7 Stack 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.BETA6/ss7
[usr]$ ant deploy
```

To undeploy these services

```
[usr]$ cd ss7-1.0.0.BETA6/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.

3.1.3. Running Mobicents SS7 Stack

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

3.1.3.1. 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.BETA6 for the first time, there are no `Linkset` defined. You need to use Shell Client to connect to ss7-1.0.0.BETA6 as defined in *???* and create `Linkset` as per your need and SS7 card installed

Once the `Linkset` are defined, the state and configuration of `Linkset` and `Link` is persisted by `LinksetManager` which stands server re-start.

3.1.3.2. Stopping

You can shut down the server(s) you can run 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
Shutdown complete
Halting VM
```

3.2. Configuring Mobicents SS7 Stack

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.

3.2.1. Configuring LinksetFactory

Concrete implementation of `LinksetFactory` is responsible to create new instances of corresponding `Linkset` when instructed by `LinksetManager`. Mobicents SS7 Stack defined three linkset factories :

- `DahdiLinksetFactory`

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

- `DialogicLinksetFactory`

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

- `M3UALinksetFactory`

```
<bean name="M3UALinksetFactory"
  class="org.mobicents.ss7.m3ua.oam.M3UALinksetFactory">
</bean>
```

Its highly unlikely that you would require all three factories on same server. If you have `dahdi` based SS7 card installed, keep `DahdiLinksetFactory` and remove other two. If you have `dialogic` based SS7 card installed, keep `DialogicLinksetFactory` and remove other two. If you don't have any hardware installed and depend on M3UA, keep `M3UALinksetFactory` and remove other two. Nevertheless you can still use above factories in any combinations.

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

3.2.2. Configuring LinksetManager

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

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

```
    </bean>
```

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

> ⚠️ **Warning**
>
> `linksetmanager.xml` should never be edited by hand. Always use Shell Client to connect to Mobicents SS7 Stack and execute commands.

3.2.3. Configuring ShellExecutor

`ShellExecutor` is responsible for listening to incoming command. Received commands are executed on local resources to perform actions like creation of `SCCP` routing rule.

```xml
<bean name="ShellExecutor" class="org.mobicents.ss7.ShellExecutor">
  <property name="address">${jboss.bind.address}</property>
  <property name="port">3435</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.

> ℹ️ **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]

3.2.4. Configuring SS7Service

`SS7Service` acts as core engine binding all the components together. To get holistic view of SS7 Service look at *Section 2.3, "SS7 Service"*

```xml
<bean name="SS7Service" class="org.mobicents.ss7.SS7Service">
```

rectly=true)</

annotation>

```
      <property name="jndiName">java:/mobicents/ss7/sccp</property>
            <property name="configPath">${catalina.home}/deploy/mobicents-ss7-service/sccp-
routing.txt</property>
      <property name="linksetManager"><inject bean="LinksetManager" /></property>
      <property name="shellExecutor"><inject bean="ShellExecutor" /></property>
   </bean>
```

SS7 service creates new instance of SCCP stack and binds it to JNDI address `java:/mobicents/ss7/sccp`. The JNDI name can be configured to any valid JNDI name specific to your application.

`configPath` is configuration path for SCCP Stack. To know more about SCCP configuration refer *Chapter 6, SCCP*

`linksetManager` property holds the reference to `LinksetManager` bean

`shellExecutor` property holds the reference to `ShellExecutor` bean. `SS7Service` creates link between `ShellExecutor` and `LinksetManager` such that all commands for linkset are directed to `LinksetManager`.

3.3. 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. you will need 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*

3.3.1. 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.BETA6.

   [usr]$ svn co http://mobicents.googlecode.com/svn/tags/protocols/ss7/ss7-1.0.0.BETA6

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 available "dahdilinux" and "dialogiclinux" appart from default profile that doesn't try to compile the native modules.

   > **i** **Note**
   >
   > Native modules are supported only for linux OS for now.

   Use "dahdilinux" profile if linux server on which this code is built already has dahdi module installed. Make sure you pass "include.zap" system property pointing to correct directory where dahdi is installed

   ```
   [usr]$ cd ss7-1.0.0.BETA6
   [usr]$ mvn install -Pdahdilinux -Dinclude.zap=/usr/include/dahdi
   ```

   Use "dialogiclinux" profile if linux server on which this code is built already has dialogic module installed. Make sure you pass "include.dialogic" system property pointing to correct directory where dialogic libraries are installed

   ```
   [usr]$ cd ss7-1.0.0.BETA6
   [usr]$ mvn install -Pdialogclinux -Dinclude.dialogic=/usr/include/dialogic
   ```

   To build Mobicents SS7 Stack without building any native libraries use

   ```
   [usr]$ cd ss7-1.0.0.BETA6
   [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.BETA6/release
[usr]$ ant
```

3.3.2. Development Trunk Source Building

Similar process as for *Section 3.3.1, "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 `dahdi` based SS7 cards like `diguim` and `sangoma`. Generally `dahdi` based SS7 crads doesn't have MTP2/MTP3 support on board and relies on external software to provide these services.

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

4.1. Sangoma

To install Sangoma cards visit the Sangoma wiki at *http://wiki.sangoma.com/*

4.2. Diguim

To install Diguim cards visit the Diguim site at *http://www.digium.com/en/products/digital/*

4.3. Dialogic

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

Shell Command Line

5.1. 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 `Linksets` .

5.2. 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 5.1. Connec to remote machine

```
mobicents>ss7 connect 10.65.208.215 3435

mobicents(10.65.208.215:3435)>
```

> **i** **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 5.2. Disconnect

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

Bye
mobicents>
```

5.3. Linkset Management

Linksets are managed by `linkset` command. It allows to perform following:

- create linkset

- delete linkset

- activate linkset

- deactivate linkset

- create link

- delete link

- activate link

- deactivate link

- list state of linksets and present links

5.3.1. Create Linkset

Linkset can be create by issuing command with following structure:

```
linkset create <linkset-type> opc <point-code> apc <point-code> ni <network-id> <linkset-name>
```

or in case of dialogic:

```
linkset create dialogic opc <point-code> apc <point-code> ni <network-id> srcmod <src-mode>
 destmod <dest-mode> <linkset-name>
```

Where:

linkset-type
> refers to type of linkset to be created, ie. `dahdi` , `dialogic` or `m3ua` . Correct values depend on which linkset factories have been deployed.

point-code
> is simply `MTP` point - either local(`opc`) or remote(`dpc`)

ni
> is simply network identifier. It can have following values:

> 0
>> International network

> 1
>> Spare (for international use only)

> 2
>> National network

> 3
>> Reserved for national use

linkset-name
> simple string name, which identifies linkset

Example 5.3. Linkset creation

```
mobicents(10.65.208.215:3435)>linkset create dahdi opc 1 apc 2 ni 0 linkset1
LinkSet successfully added
mobicents(10.65.208.215:3435)>linkset create dialogic opc 3 apc 4 ni 3
 srcmod 1 destmod 2 linkset2
LinkSet successfully added
```

5.3.2. Remove Linkset

Linkset can be deleted by issuing command with following structure:

```
linkset delete  <linkset-name>
```

Where:

linkset-name
    is name set during link creation

Example 5.4. Linkset Removal

```
mobicents(10.65.208.215:3435)>linkset delete linkset1
LinkSet successfully deleted
```

5.3.3. Activate Linkset

Linkset can be activated by issuing command with following structure:

```
linkset activate  <linkset-name>
```

Where:

linkset-name
    is name set during link creation

Example 5.5. Linkset Activation

```
mobicents(10.65.208.215)>linkset activate linkset1
LinkSet activated successfully
```

23

5.3.4. Deactivate Linkset

Linkset can be deactivated by issuing command with following structure:

linkset deactivate <linkset-name>

Where:

linkset-name
    is name set during link creation

Example 5.6. Linkset Deactivateion

```
mobicents(10.65.208.215)>linkset deactivate linkset1
LinkSet deactivated successfully
```

5.3.5. Create Link

Link can be created in Linkset by issuing command with following structure:

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

Where:

span-num
    integer number. It represents port number in card(indexed from 0).

code-num
    link code(sls assigned to this link).

channel-num
    integer number indicating time slot number(TDM time slot).

linkset-name
    is name set during link creation.

link-name
    name which identifies link in linkset.

Example 5.7.

```
mobicents(10.65.208.215:3435)>linkset link create span 1 code 1 channel 1
 linkset1 link1
Link successfully added
```

5.3.6. Remove Link

`Link` can be removed from in `Linkset` by issuing command with following structure:

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

Where:

linkset-name
    is name set during link creation

link-name
    name which identifies link in linkset

Example 5.8. Link Removal

```
mobicents(10.65.208.215:3435)>linkset link delete linkset1 link1
Link successfully deleted
```

5.3.7. Activate Link

Link can be activated by issuing command with following structure:

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

Where:

linkset-name
    is name set during link creation

link-name
    name which identifies link in linkset

Example 5.9. Link Activation

```
mobicents(10.65.208.215:3435)>linkset link activate linkset1 link1
Link activated successfully
```

5.3.8. Deactivate Link

Link can be deactivated by issuing command with following structure:

```
linkset link deactivate <linkset-name> <link-name>
```

Where:

linkset-name
    is name set during link creation

link-name
    name which identifies link in linkset

Example 5.10. Link Deactivateion

```
mobicents(10.65.208.215:3435)>linkset link deactivate linkset1 link1
Link deactivated successfully
```

5.3.9. Show status

Linkset and Link's status can be viewed by issuing command with following structure:

```
linkset show
```

Example 5.11. Linkset Status

```
mobicents(10.65.208.215:3435)>linkset show
linkset1     dahdi    opc=1          apc=2          ni=0
 state=UNAVAILABLE
    link1      span=1   channelId=1   code=1  state=UNAVAILABLE
```

The possible state of Linkset are

- UNAVAILABLE : Indicates the linkset does not have any "available" links and cannot transport traffic

- SHUTDOWN : Indicates the linkset has been shutdown in the configuration

- AVAILABLE : Indicates the linkset has at least one available link and can carry traffic

The possible state of Link are

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

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

- AVAILABLE : Indicates the link is active and able to transport traffic

- FAILED : A link is FAILED when the link is not shutdown but is unavailable at layer2 for some reason. For example Initial Alignment failed or the link test messages sent by MTP3 are not being acknowledged.

5.4. SCCP Management

Currently Shell does not support SCCP management.

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 and more flexible methods of routing.

6.1. Routing Management

Currently `Shell` does not support commands to manage `SCCP` routes.

6.2. Routing Configuration

Routing rules are persisted within file. By default its `sccp-routing.txt`. This file is managed by `SCCP` stack and `SS7 Service`, it is not encouraged practice to edit this file by hand. However for complete rerefence, you can find structure of this file below:

```
sequence;pattern;translation;mtpinfo
```

> **Note**
>
> Fields within `pattern`, `translation` and `mtp`, separated by `#`, empty value is indicated by single `space`. Each part of routing entry is separated by `;`.

sequence
: is simple sequence number, each entry increases it by 1.

pattern
: is simply `SCCP` like regular expresion. Destination address of message is matched against this to check which rule should be triggered.

Table 6.1. pattern content

Name	Type	Description
translation type	java.lang.Integer	Network specific ID which determines how global title analysis is performed.

Name	Type	Description
numbering plan	enum	Determines which numbering plan is used for global title. It can have one of following values: UNKNOWN, ISDN_TELEPHONY, DATA, TELEX, MERITIME_MOBILE, LAND_MOBILE, ISDN_MOBILE
nature of address	enum	Determines type of address. It can have one of following values: SPARE, SUBSCRIBER, UNKNOWN, NATIONAL, INTERNATIONAL
digits	java.lang.String	Simply digits of number, ie: +91 417688345892
sub-system number	java.lang.Integer	local subsystem number, used to route between local services in SS7, ie. between HLR and VLR.

translation

    has the same structure as `pattern` . If `pattern` matches, destintion address is changed to `translation` .

mtpinfo

    `mtpinfo` indicates which link should be chosen when routing to remote location. In case routing is performed localy, its not present.

Table 6.2. mtpinfo content

Name	Type	Description
name	java.lang.String	
adjacent point code	java.lang.Integer	point code of remote, mtp link.
origin point code	java.lang.Integer	polint code of local, mtp link.
signaling link selector	java.lang.Integer	number indicating TDM multiplexed link.

Example file entry looks as follows:

Example 6.1. sccp-routing.txt example entry

0; #ISDN_MOBILE#NATIONAL#9023629581# ; #ISDN_MOBILE#INTERNATIONAL#79023629581# ;linkset#14083

sequence
   is set to 0, since its only rule in a file(or first).

pattern
   matches natioan mobile number with following digits: `9023629581`

translation
   changes destination to international mobile with following digits: `79023629581`. Since no sub-system number is present, this rule requires `mtpinfo` to indicate target.

mtpinfo
   indicates link through which message will be sent to next hop. Link:

   * Belongs to linkset with name equal to `linkset`.

   * Adjacent point code is equal to `14083`.

   * Origin point code is equal to `14155`.

   * Signaling link selector is equal to `0`. Note that this value may be overriden by transport layer.

6.3. Mobicents SS7 Stack SCCP Usage

The `org.mobicents.protocols.ss7.sccp.SccpStack` is responsible for taking the config file and turning it into `org.mobicents.protocols.ss7.sccp.Router`. All the sccp messages sent bu SCCP User Part are routed as per the rule configured in Router

The SCCP User Part gets handle to `SccpStack` by doing JNDI look-up as explained in *Section 6.4, "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 `org.mobicents.protocols.ss7.sccp.parameter.SccpAddress`. For every incoming SccpMessage, if the called party address matches with this local SccpAddress, 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.

The UML class diagram looks like

Mobicents SS7 Stack SCCP Class Diagram

6.4. 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 3.2.4, "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();
      }
    }
```

6.5. SCCP User Part Example

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

```java
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);
```

```java
      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);
```

```
            }

        }

    }
```
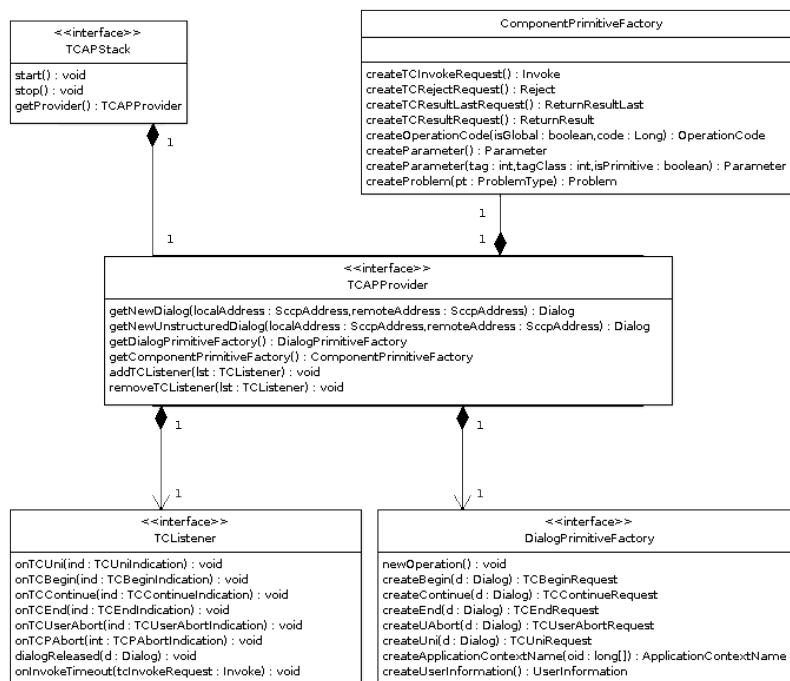
TCAP

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

`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 registers `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

`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 listenr. `TCAP` stack creates internaly Mobicents `MAP` Stack implementation. Passed `SccpAddress` is used to match against incoming messages destination address.

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

    TCAPStack tcapStack = new TCAPStackImpl(sccpPprovider, localAddress);


    ...

    private SccpAddress createLocalAddress()
    {
        int translationType = 0;

        int subSystemNumber = 0;


        GlobalTitle gt = GlobalTitle.getInstance(translationType,

        NumberingPlan.ISDN_MOBILE, NatureOfAddress.NATIONAL, "1234");

        SccpAddress localAddress = new SccpAddress(gt, 0);
        return localAddress;
    }
```

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 6.4, "Access Point"*

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

```java
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());
invoke.setOperationCode(cpFactory.createOperationCode(true,new Long(12)));
//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 };
```

```java
    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());

      invoke.setOperationCode(cpFactory.createOperationCode(true,new Long(12)));
      //no parameter
      this.clientDialog.sendComponent(invoke);

      ApplicationContextName acn = this.tcapProvider.getDialogPrimitiveFactory()
         .createApplicationContextName(_ACN_);
```

```java
      //UI is optional!
      TCBeginRequest tcbr = this.tcapProvider.getDialogPrimitiveFactory().createBegin(this.clientDialog);
      tcbr.setApplicationContextName(acn);
      this.clientDialog.send(tcbr);
   }

   public void dialogReleased(Dialog d) {


   }

   public void onInvokeTimeout(Invoke tcInvokeRequest) {

   }

   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) {
```

```java
        // TODO Auto-generated method stub

    }

    public void onTCUserAbort(TCUserAbortIndication ind) {
        // TODO Auto-generated method stub

    }


    public static void main(String[] args)
    {

        int translationType = 0;
        int subSystemNumber = 0;
        GlobalTitle gt = GlobalTitle.getInstance(translationType,NumberingPlan.ISDN_MOBILE,NatureOfAddress.NAT
        SccpAddress localAddress = new SccpAddress(gt,0);

        gt = GlobalTitle.getInstance(translationType,NumberingPlan.ISDN_MOBILE,NatureOfAddress.NATIONAL, "15
        SccpAddress remoteAddress = new SccpAddress(gt,0);
        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.

8.1. SS7 Stack MAP Usage

`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. The class diagram looks like
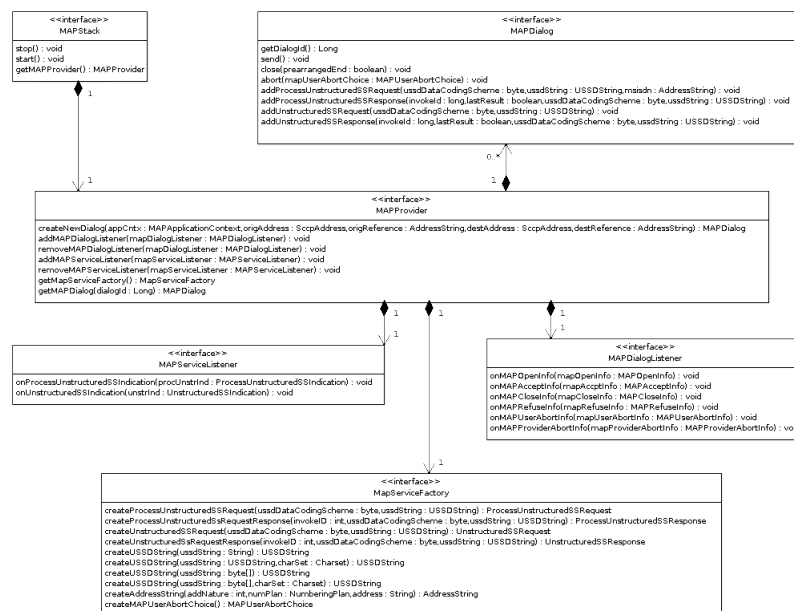
Mobicents SS7 Stack MAP Class Diagram

> **i** **Note**
>
> Mobicents SS7 Stack MAP has implementation for USSD Message only. Any
> contribution to implement message specific to SMS or other are welcome. We will
> provide you all the help that you may need initially.

`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 new instance of `SccpAddress` representing address to which bind listener.
This addess will be used to match against destination address in `SCCP` messages.

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

    MAPStackImpl mapStack = new MAPStackImpl(sccpPprovider, localAddress);

    ...

    private SccpAddress createLocalAddress()
    {
        int translationType = 0;

        int subSystemNumber = 0;


        GlobalTitle gt = GlobalTitle.getInstance(translationType,

        NumberingPlan.ISDN_MOBILE, NatureOfAddress.NATIONAL, "1234");

        SccpAddress localAddress = new SccpAddress(gt, 0);
        return localAddress;
    }
```

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 6.4,
"Access Point"*

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

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

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

```java
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 `MAPProvider` to create new `MAPDialog` and send USSD message

```java
// First create Dialog
MAPDialog mapDialog = mapProvider.createNewDialog(
    MAPApplicationContext.networkUnstructuredSsContextV2,
    destAddress, destReference, origAddress, origReference);

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, MAPServiceListener {

    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, NumberingPlan.ISDN,
            "31628968300");
```

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

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

  mapProvider.addMAPDialogListener(this);
  mapProvider.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() {
  GlobalTitle gt = GlobalTitle
      .getInstance(
          NatureOfAddress.NATIONAL.getValue(),
          org.mobicents.protocols.ss7.indicator.NumberingPlan.ISDN_MOBILE,
          NatureOfAddress.NATIONAL, "1234");

  return new SccpAddress(gt, 0); // 0 is Sub-System number
}

private static SccpAddress createRemoteAddress() {
  GlobalTitle gt = GlobalTitle
      .getInstance(
          NatureOfAddress.NATIONAL.getValue(),
          org.mobicents.protocols.ss7.indicator.NumberingPlan.ISDN_MOBILE,
          NatureOfAddress.NATIONAL, "1572582");
```

```java
        return new SccpAddress(gt, 0); // 0 is Sub-System number
    }

    public void run() throws Exception {

        // First create Dialog
        MAPDialog mapDialog = mapProvider.createNewDialog(
                MAPApplicationContext.networkUnstructuredSsContextV2,
                destAddress, 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 onMAPAcceptInfo(MAPAcceptInfo mapAccptInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPCloseInfo(MAPCloseInfo mapCloseInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPOpenInfo(MAPOpenInfo mapOpenInfo) {
        // TODO Auto-generated method stub
```

```
    }

    public void onMAPProviderAbortInfo(MAPProviderAbortInfo mapProviderAbortInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPRefuseInfo(MAPRefuseInfo mapRefuseInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPUserAbortInfo(MAPUserAbortInfo mapUserAbortInfo) {
        // TODO Auto-generated method stub

    }

    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();

    }

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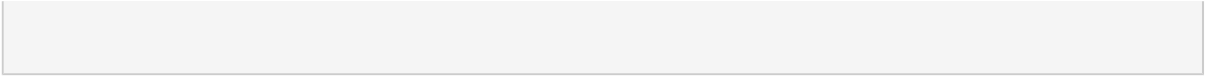
Appendix A. Java Development Kit (JDK): Installing, Configuring and Running

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"
```

> **i** **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 `export`ed 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`.

> **i** **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/
```

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