

# Signaling Manager CLI User Guide

## DESCRIPTION

**Copyright**

© Ericsson AB 2009, 2016-2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	General	1
1.2	Starting the CLI	2
<b>2</b>	<b>The User Interface</b>	<b>3</b>
2.1	Shell Features	3
2.2	CLI Naming Conventions	4
2.3	Modifying MO Key Attributes	4
2.4	The Help System	5
2.5	Modes of Operation	6
2.6	Standards in Signaling Manager	6
2.7	File Versions in Signaling Manager	7
<b>3</b>	<b>Common Use Cases</b>	<b>9</b>
3.1	Creating Configuration	9
3.2	Activating Configuration	9
3.3	Process Handling	10
3.4	Monitoring	11
3.5	Migrating CLI Commands	12
<b>4</b>	<b>Step-by-step Example</b>	<b>15</b>
	<b>Reference List</b>	<b>17</b>





# 1 Introduction

The Signaling Manager provides a Graphical User Interface (GUI) and a Command Line Interface (CLI) for the configuration and operation of the Signaling stack. This document describes only the CLI, the GUI is described in a separate User Guide, see Reference [2]. The configurations are stored in the local file system or in a remote file system using built-in support for (S)FTP. The Signaling Manager CLI user information is available as this document, the help command and in CLI command descriptions.

It is highly recommended to read through this document before using the system. The instructions explain how to work efficiently with the CLI.

Signaling Manager reads its own configuration file `signmgr.cnf` at start up. Here you can configure the system standard to use, file locations, remote file access, and so on. A detailed description of the parameters in this file is available in the Configuration File Description, see Reference [1]. An overview of the available parameters is listed if the help option is used: **signmcli -h**.

It is also possible to edit the Signaling Manager configuration file from the tool itself using the **smsmc** command.

It is very important that the tool is correctly installed for proper operation and interaction with the Signaling stack.

## 1.1 General

The CLI is full featured in the sense that all functionality of Signaling Manager is available through commands.

All supported functionality can roughly be divided into the following groups

- Create and modify Configuration.

When configuring from the CLI, the principle is based on creating and modifying MO instances in a tree structure. For each MO class there are one command of each of: Initiate (create) , Change, End (delete) and Print.

- Stack process management

Monitoring, adding, restarting and removing stack processes.

- Invoking Actions or Statistics.

All actions are tied to a specific MO.

- Monitoring alarms and stack status.

- Basic Signaling Manager functions.



Miscellaneous commands such as creating, importing, opening, saving and exporting the configurations.

## 1.2 Starting the CLI

The CLI is started using the **signmcli** script whether it shall be run in batch mode or in interactive mode. In order to start in batch mode, pipe the file with CLI commands to the script.

Interactive mode: **signmcli**

Batch mode: **signmcli < <batch file> or cat <batch file> | signmcli**, where <batch file> is a file containing all the commands.

**Note:** It is also possible to load batch files from within the CLI using the **loadbatch** command.

To be able to use the features of the interactive CLI shell it must be started in a terminal that supports VT100 escape sequences. Most UNIX terminals have this support. When working in mixed UNIX environments with the CLI process running in e.g. Solaris while the terminal is running in Linux there is a small incompatibility in the delete key scan codes between the systems that can be compensated with the `delete.keys` parameter.

All Signaling Manager configuration parameters can be set both on the command line and in the configuration file. The command line overrides the configuration file settings.

The parameter `online` decides if the tool will act like a standalone configuration file editor or as a manager of the stack processes.

If the configuration parameter `online` is set to yes, the tool will try to open the active or planned configuration and then connect automatically to the stack by issuing the **connect** command.

If the parameter `online` is set to no, none of the above will be done. When operating in this mode it is not possible to connect to the stack.



## 2 The User Interface

### 2.1 Shell Features

When starting the CLI in interactive mode in a terminal that supports VT100 escape sequences the shell provides a set of useful features commonly available in standard UNIX shells.

<b>Completion</b>	Tab key can be used to complete your typed input to get the possible matches. If only one single match is found, you can still press Tab key to complete the command with possible arguments. If the selected argument has a predefined set of values, by pressing the Tab key, the possible values will be listed to select from.
<b>History</b>	Up and Down arrows keys are used for command history to go back to the previous or to the next executed commands.
<b>Editing</b>	Left and Right arrow keys together with Backspace and Delete works as expected in an editor. Moreover, Ctrl-A can be used to move to the start of the line and Ctrl-E to move to the end. Ctrl-K can be used to cut out everything from the cursor to the end of line, and Ctrl-Y to paste it back again.

The completion feature is very useful if it is used correctly. The following step by step example illustrates how it works:

1. Type **m3** and press tab. All the commands for the M3 module are listed together with their corresponding full names. This makes it easier to find the right command.
2. Type the rest of the characters in the chosen command **lsi** and press tab. The completion will now find the command, find the first mandatory parameter and print out M3LSI: LN0=.
3. Choose a linkset number and add a comma (**1,**) and press tab. Note that the comma is important to show that the value has been fully entered. The completion will now printout the next mandatory parameter M3LSI: LN0=1, SLC=.
4. Choose an SLC value, add a comma and press tab just as in the previous step. The completion will now printout a list of all the parameters that may be entered together with their full names if they are abbreviated.
5. Type **i** and press tab and fill in the instance ID value **0,,** the output is now M3LSI: LN0=1, SLC=1, IID=0,



6. Type **tr** and press tab, the trunk parameter will be added. Press tab once more. Now all the valid values for the trunk parameter will be listed. Both the names and the numbers are valid. So choosing **2** is the same as choosing **PCMA**.

## 2.2 CLI Naming Conventions

The majority of Signaling Manager CLI commands consist of five characters, although exceptions from this rule exist when it comes to basic management functions such as open and save configuration files. Also, actions and statistics commands commonly use more letters to become unique. In order to understand these five character commands, some general naming conventions apply.

- The first two characters reflects the module to which the command applies.

Examples: **SS** is the initial part of commands for Signaling System, **TCAP** commands start with **TC** and **SCCP** commands start with **SC**.

- The next two characters identifies which MO the command operates on.
- The last character reflects the operation. To achieve similarities with older applications related to Signaling Manager functionality, commands used for creating things ends with **I** (for initialize). The same reasons apply to modify and delete operations. Modification commands ends with **C** (for change) while delete operations ends with **E** (for end). Commands printing information ends with a **P**.

Examples:

**TCAPI** - creates a TCAP configuration

**TCAPC** - modifies an existing TCAP configuration

**TCAPE** - deletes an existing TCAP configuration

**TCAPP** - prints the current TCAP configuration

## 2.3 Modifying MO Key Attributes

If there is a need to modify the value of MO key attribute use the following command format:

**<command name>:<attribute name>=<current attribute value>, New<attribute name>=<new attribute value>;**

For example the command **M3RCONTC:RC=3, NewRC=4;** modifies a Routing Context for the Local AS from 3 to 4.





## 2.4 The Help System

A complete online help-system is available through the help-command. Enter the command **help: cmd=help;** to get more information about this. The core set of commands can be found in the SignalingSystem MO, list them using **help:GRP=SignalingSystem;**. The descriptions for the commands/parameters depend on which signaling standard that is configured, see Section 2.6 on page 6 and on the expert mode setting, see Section 2.5 on page 6.

The following notations are used when a command is explained using the help on a command:

Table 1 CLI notations

Notion	Name	Description
[ , ]	Optional — all or none	It means that it contains optional arguments due to [ ] notation and either every one of the parameters specified within [ ] separated by comma must be defined or none of them in order to be able to perform the command.  Example: [IID,TR,TS] in the M3SLI:LN0,SLC,[LPC],[LPC],[IID,TR,TS] command.
( , )	Required group	It means every one of the parameters within the parentheses separated by comma must be defined in order to be able to perform the command.  Parentheses make the parameters as a group to indicate that the specified group is needed when the group is used within other notation, for example as { ( , )   (.) }
{   }	One of	It means that one of the arguments within { } separated by   must exist in order to be able to perform the command.
{[ ],[ ]}	At least one	It means that at least one of the arguments identified in [ ] within { } and separated by comma must exist in order to be able to perform the command.
[{   }]	At most one	It means that it contains optional arguments and due to { } one of the arguments within { } separated by   must exist to perform the command.



For a complete listing of all the available commands you may use the **CLI Command Descriptions** in the Signaling Manager reference documentation, see Reference [3].

## 2.5 Modes of Operation

Signaling Manager has a set of modes that controls its behavior, they are there to guard the configuration against unwanted modification and to reduce the amount of visible parameters. The command **modeprint**; shows the current status of the configuration and expert modes. They are modified by the **confmode:mode=<mode>**; and **expert:mode=<mode>**; respectively.

The Configuration Mode is used to guard the configuration against unintentional changes:

Table 2 Configuration Modes

Initial (init)	This mode is used when all types of changes shall be allowed. When the changes shall be applied to the stack the configuration files are prepared and the stack is started or restarted.
Reconfiguration (reconf)	This is the default mode and only allows changes that can be applied without restarting the stack.
Readonly	Locks all properties from change. This mode is good for monitoring or when more than one user is accessing the stack simultaneously. When in read only mode no lock file is created.

The expert mode is just used to control the visibility of properties. The default is to be in “basic” mode, i.e. expert mode is off. In this mode only the commonly edited properties are shown in the printouts. It is still possible to edit all properties and the completion feature still completes against all available properties but the online help (**help: cmd=<cmd>**;) and printouts will just show the basic properties.

## 2.6 Standards in Signaling Manager

The **standard** setting controls which standard the CLI help will describe and which standard that will be the default system standard. More information on how to set it is found in Configuration File Description, see Reference [1].

An overall system standard is set in Signaling Manager configuration included in `signmgr.cnf` or as a command line option. However, elements on different stack configuration levels such as network level, can be assigned specific standards overriding the system standard. The default value is though a reference to the system standard value.

If standard is not applicable on system level, for example in a gateway configuration between networks of different standards, the standard in



`signmgr.cnf` can be assigned a NA value. This value is applicable for system standard only, not for standard parameters on other configuration levels. If system standard is set to NA, standard values must be set (possible in several places) on a more detailed configuration level to provide all parts of the configuration with a valid standard value.

Changing the value of any occurrence of standard will change the operating conditions for Signaling Manager in the scope of the standard value. In some cases, the user interface is not affected, but in other cases the user is able to see a difference apart from the actual standard value. For example, default values might be changed as well as contents of various lists and descriptions.

## 2.7 File Versions in Signaling Manager

For some modules, Signaling Manager has support for generation (export) of more than one CNF file version. When this is the case the appropriate file version that correspond to the module version has been configured in the settings for Signaling Manager. The properties or relations that are not applicable to the selected file version is automatically hidden in the CLI. It is however not possible to hide information about these in the documentation that is included with Signaling Manager, so if the documents mentions elements that are not visible then they are not applicable for your Signaling Stack.





## 3 Common Use Cases

### 3.1 Creating Configuration

The CLI operates on the same Information Model as the GUI. The Information Model is tree based and thus the configuration tree must be built from the root element and down. Each MO (Managed Object) in the tree has at least four commands associated as explained in Section 2.2 on page 4. When creating a new configuration it is mainly the Initiate and Change commands that are used. The initiate command usually need one or more parameters to identify the parent element that the new element should be added to, and one or more parameters which will be used to identify the new element. The initiate command also prints out the settings of the new element in the form of a initiate/change command pair. This can be used to determine if some setting need to be changed. To simplify configuration process it is possible to use commands copy and paste. If you need to create MO with the same parameters as an existing MO, use **copy: MOPath="//<ClassIdentifier>[<conditions>]"**. This will copy MO, specified by MOPath. Then **paste: MOPath="//<ClassIdentifier>;"** to paste copied MO in a specified path. For more information about MO Paths, please see the Signaling Manager User Guide (Reference [2])

It is recommended to use the Information Model pictures in the Signaling Manager reference documentation, see Reference [3]. They can be used to understand the hierarchical structure to see which child elements to add to which parent. Another way to understand the order in which the commands should come is to open an example configuration using **open: file=<filename>;** and execute the command **printall;**, this will generate the commands necessary to create the loaded configuration.

The **validate;** command is useful to get feedback on the configuration, it will point out any errors in the configuration. For each error a change command is usually printed that suggests where to correct the error.

### 3.2 Activating Configuration

The configuration activation that is done with the command **configure** performs quite a lot of tasks to make sure that the stack configuration is consistent and that it is always clear which configuration that is currently active (used by the stack). To avoid that the currently active configuration is overwritten during configuration changes the **save** command never saves to `active.om.cim` but will automatically save to `planned.om.cim` instead. It is just the **configure** command that updates the active configuration.

**Note:** Do not do manually copy files to the name `active.om.cim` as this will circumvent the consistency checks and will make the reconfiguration checks incorrect as the active file isn't what is actually used in the stack.



### 3.2.1 Activating an Initial Configuration

When creating a configuration for a new stack, or when it is not important to keep the stack running while applying the configuration then the configuration is applied in the following way:

1. Edit the configuration, use **confmode: mode=init;** to remove any change restrictions. Make sure it pass the validation.
2. Prepare the configuration files using **configure: initial;**
3. Either start the stack it is not running, or use **procr: all;** to restart it. The stack will automatically initialize itself with the new configuration.
4. If Signaling Manager was connected to the stack before the restart it will automatically reconnect, otherwise use **connect;** and use **procp;** to printout the status of the stack. When it has started everything all processes should be Running.

### 3.2.2 Activating a Configuration in Runtime

Make sure start with the configuration that is currently in use by the stack. When starting Signaling Manager in online mode this configuration is loaded automatically. The configuration mode must be reconf/Reconfiguration. This is the default and makes sure that only properties that are allowed to change without affecting the traffic are editable..

1. Edit the configuration and make sure to stay in reconfiguration mode.
2. Reconfigure the stack using **configure;**
3. Use **procp;** to see that all stack processes have been reconfigured.

## 3.3 Process Handling

It is possible to monitor and operate the stack processes through the Signaling Manager proc commands. It is necessary to be connected with the stack to use them. Use the command **connect;** to become online. (This command is not available if operating in offline mode). When connected the **procp;** command is used to list the status of all stack processes.

```
cli> procp;
Process                               State
BE_RP:1 [seksux039]                  Running
FE_MTPL2_ISR:0 [seksux039]           Running
BE_NMP:0 [seksux039]                  Running
OAM:0 [seksux039]                     Running
```

Example 1 Output from procp

The information in the list is on each row <process class>:<instance id> [<host name>] <state>. This is very useful to see that the system is up and



running. That a process is running does however only say that it is in a state where it is able to handle traffic. For a finer granularity in the status, see Section 3.4 on page 11.

Table 3 Process states

Down	The process is expected to exist according to configuration but it has not been started. This is typically a temporary state. If it is stable the host or ecp process is not up.
Unknown	There is no connection with the stack so the status of the process can't be determined.
Idle	The process has been started but has not yet received any configuration. Normally OAM makes sure that this state is only temporary. If it is stable then the configuration is probably either missing or faulty.
Initialized	The process has read its configuration successfully. Normally OAM makes sure that this state is only temporary. If it is stable then the conditions for start have not been met. E.g. a BE process refuse to start if there are no running FEs.
Running	The process is ready to handle signaling traffic. This state is the stable state which all processes should be in.
Reconfigured	Same as Running, but indicates that a new configuration was successfully read in runtime.

The other process related commands are:

Table 4 Process commands

procr: all;	Restart the entire stack.
procr: ity=<type>, iid=<id>;	Restart a single process.
proci: host=<host>, iid=<id>, ity=<type>, grp=<group>;	Create/Initiate a new process. The host must be one of the configured hosts listed by <b>echsp</b> ; and the start group must be one of those listed by <b>ecpgp</b> ;
proce: ity=<type>, iid=<id>;	Remove/End a process.

## 3.4 Monitoring

To use Signaling Manager for monitoring and controlling a running stack it is necessary to start it in online mode (which is default). In this mode Signaling Manager will automatically connect to the stack and open the `active.om.cim` file that contains the configuration currently in use by the stack. It will now be properly prepared for monitoring, since the configuration in Signaling Manager match the configuration used by the stack. It is not recommended to perform monitoring while changing the configuration as the configuration in Signaling



Manager and the stack will then differ. In this state it will only be possible to correctly monitor the parts which have not changed.

A good way to get an overview of the status of a core set of elements in the stack is using the System Overview. Use the **sysop;** command to get a printout of the status tables. If the System Overview is disabled it can be enabled with **SMSMC:SystemOverviewEnabled=yes;**. This option is stored in `signmgr.cnf`.

To see the currently active alarms the **alrmp;** command is used. The notifications can not be viewed from within the CLI, they are instead logged to the `sm.alarm.log` file. To turn this logging off, you can set `alarm.log.quantity` to 0 (zero).

Signaling Manager will by default listen to incoming alarms from the stack processes. Which alarms Signaling Manager will listen to are set in the alarm mask configuration of the OAM IMC. change it using the **modc:proctype=<pt>,userid=<uid>,AlarmMask=<mask>;** command.

**Note:** Signaling Manager subscribes, by default, to all ECM alarms (i.e. stack process related). These can be enabled or disabled by setting the `ecm.Alarm.enabled` property to `no` in the Signaling Manager configuration. The stack does not need to be restarted in order to activate this modification.

## 3.5 Migrating CLI Commands

### 3.5.1 General

There is no automatic migration function for CLI commands. If deviations are detected in stored CLI commands after an upgrade of Signaling Manager, these must be adjusted manually by editing the file containing the command set. It is recommended to, whenever possible, store configurations in CIM format and let the automatic migration do the work.

Differences in parameters between different Signaling Manager versions affect the CLI if the modified parameters are used in a CLI command. Thus, batch files containing sets of CLI commands must be migrated to reflect the new parameter set for each affected command.

If a single CLI command does not fulfill the new required syntax, an error message will be returned and the command will not be executed. In case several commands are run in a batch, the entire set of commands will be analyzed for migration changes prior to execution of the first command. This is to avoid inconsistency caused by failing commands late in the command sequence.

### 3.5.2 Command Analyzer

To be able to verify the syntax of old files containing sets of CLI commands when migrating from one Signaling Manager version to another, the CLI Command





Analyzer can be invoked without actually executing the commands. If there are deviations, the output is a list of all deviations in the provided set of commands.

The Command Analyzer is invoked separately by starting the CLI with the configuration parameters `online=NO` and `batch.analysis=YES`. See Configuration File Description (Reference [1]) for configuration parameter details.

A single command set file is analyzed by running the CLI in batch mode. If several files are to be tested or if the environment puts restrictions on command line options and redirection, an alternative is to start the CLI once in interactive mode and execute the **loadbatch** CLI command for each file. Starting the CLI in different modes is described in Section 1.2 on page 2.





## 4 Step-by-step Example

To illustrate how the CLI works and show a practical example of how a configuration is built from scratch. To keep the example small only MTP-L2 and M3 will be covered. This shows the principles that can, using the suggestions in Section 3.1 on page 9 be used to create any configuration. The default values have been used for all non-mandatory parameters, change commands are needed to edit these less frequently used parameters.

Create a new configuration, this creates a new system root element.

```
cli> new;
```

Create MTP-L2 ISR front end, supplying the necessary settings in the init command. (Note that it should be a single line)

```
cli> L2IFI:IID=0, Bandwidth=nb, DigitalTransmissionSystem=e1,
      ClockSource=PCMA, Stand-ByClockSource=FREERUNNING;
```

Create a Trunk, which represents the physical wire connected to the board.

```
cli> L2TRI:IID=0, TLI=0, TLIM=0, TN=PCMA;
```

Create the link configuration profile which holds among other things the timer settings which are typically shared by many links.

```
cli> L2PRI:IID=0, LTID=0;
```

Create a link using the first timeslot, it automatically connects to the first (and only) configuration profile.

```
cli> L2LII:IID=0, TR=PCMA, TS=1;
```

Create the network which will hold the M3 configuration, the standard is inherited from the system standard.

```
cli> SSNWI:NWID=1, NI=NI0;
```

Create a local point code.

```
cli> SSLSI:LPC=100;
```

Create a remote point code.

```
cli> M3RPI: RPC=200;
```

Create the MTP Sign Point which holds the M3 SS7 configuration.

```
cli> M3SPI:NB="SS7 End-Point";
```

Create a linkset that use the remote point code as adjacent.



```
cli> M3LSI:LN0=1, RPC=200;
```

Create an M3 link that is connected with the MTP-L2 link.

```
cli> M3SLI:LN0=1, SLC=0, IID=0, TR=PCMA, TS=1;
```

Create a routeset to the adjacent remote point code.

```
cli> M3RSI: RPC=200;
```

Create a route in the routeset which uses the linkset.

```
cli> M3ROI:RPC=200, LN0=1;
```

Now validate and save the configuration. Note that the quotation marks are needed for any value that contain other characters than numbers or letters. Also note that the save command resolves relative paths against the configured `config.location`. So the current working directory is not used.

```
cli> validate;  
Validation was successful.  
EXECUTED  
cli> save: file="example.cim";  
EXECUTED
```



## Reference List

- [1] Signaling Manager Configuration, 19073-CNA 403 0874 Uen/
- [2] Signaling Manager User Guide, 1553-CNA 403 0874 Uen/
- [3] Signaling Manager Reference Documentation  
<installation dir>/doc/index.html