

Ericsson Command-Line Interface

INTERWORK DESCRIPTION

Copyright

© Ericsson AB 2016, 2017, 2018. 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.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	Related Information	1
1.2	Key Features of ECLI	1
1.3	Prerequisites	2
1.4	ECLI Version	3
2	CLI Concepts	7
2.1	CLI Modes	7
2.2	CLI Session	8
2.3	CLI Transaction	11
2.4	CLI Position	15
2.5	CLI Prompt	17
2.6	Context-Sensitive Help	18
2.7	Auto-Completion	23
2.8	Case Correction	29
2.9	Escaping of Special Characters	29
2.10	CLI Comments	31
2.11	Visibility Levels	31
2.12	Automatic Correction of ManagedElement ID	35
2.13	Limitations	35
3	CLI Commands	37
3.1	Summary of CLI Commands	37
3.2	Success and Error Indications	46
3.3	Display Information	48
3.4	Change and Display the Position in MO Tree	77
3.5	Change MO Attributes	79
3.6	Create MO	99
3.7	Reinitialize MO	102
3.8	Delete MO	107
3.9	MO Actions	108
3.10	Copy and Paste Configuration Data	112
3.11	Change Password Command	112
3.12	Display CLI Version	113



3.13	Pipe Utility Commands	113
3.14	Help Command	118
3.15	Deprecated Actions	119
3.16	Deprecated Options	120
3.17	CLI Commands Limitations	120
4	Terminal Properties	121
4.1	Terminal Types	121
4.2	Default Key Bindings	121



1 Introduction

This document describes the Ericsson Command-Line Interface (ECLI). It is based on industry de facto standard patterns.

The ECLI is a terminal-based command-line interface that allows the user to monitor and manage the Managed Element (ME). The ECLI enables the user to interact with the Management Information Base (MIB) through common, generic-purpose commands.

1.1 Related Information

For information on the Managed Object Model (MOM), Managed Object Classes (MOCs), Managed Objects (MOs), and related concepts mentioned in this document, refer to [Managed Object Model User Guide](#).

1.2 Key Features of ECLI

The key features of the ECLI are described in Table 1.

Table 1 Key Features of ECLI

Feature	Description
Access control	The result of the ECLI commands manipulating the MIB is subject to authentication and authorization. If the user has no permission to access an MO instance or attribute, operations behave as if the MO instance does not exist.
Auto-completion	By pressing the Tab key, all possible ECLI command completions are displayed and unique completions are added to the command line. For details, see Section 2.7 Auto-Completion on page 23.
Case correction	Auto-completed items entered by the user are automatically changed to the case defined in the MOM. For details, see Section 2.8 Case Correction on page 29.
Configurable ECLI properties	ECLI properties (command history, page break, script mode, width, and prompt configuration) can be changed for the ECLI session. For details, see Section 3.1 Summary of CLI Commands on page 37
Configuration export and import	With command show-config , the system configuration is displayed in a format that is also a valid input for the ECLI. Thus, copy/paste or terminal input/output redirection allows configuration copy. For details, see Section 3.3.8.1.4 Display Single MO and Its Child MOs in Configuration Printout Format on page 60.



Table 1 Key Features of ECLI

Feature	Description
Context-sensitive help	By pressing the ? key, a description of the ECLI command element is displayed. For details, see Section 2.6 Context-Sensitive Help on page 18.
ECLI modes	Two ECLI modes are supported. Exec mode is intended for observation and executing actions. Config mode is used for changing the ME configuration. For details, see Section 2.1 CLI Modes on page 7.
ECLI prompt	The ECLI prompt can be configured and provides information on the ECLI mode and ECLI position. For details, see Section 2.5 CLI Prompt on page 17.
Model driven	The ECLI command elements and their properties are defined in the MOM as MOCs, attributes, and actions.
Navigation	The position in the MO tree can be changed. The position determines the context of the ECLI command. For details, see Section 3.4 Change and Display the Position in MO Tree on page 77.
Scripting	ECLI scripts can be created by feeding the ECLI commands to and parsing the output from the SSH client. The SSH client buffer size (typically 4 KB) limits the amount of input and output text that can be processed.
Security	An ECLI session is running securely over SSH.
Transactions	Configuration changes are applied through atomic transactions. Thus, it is ensured that all or none of the operations are executed. For details, see Section 2.3 CLI Transaction on page 11.

Note: The examples in this document are based on models that are subject of change. For node name, NODE06ST is used as an example.

1.3 Prerequisites

This section describes the prerequisites, which must be fulfilled before using the ECLI.

1.3.1 Conditions

The following conditions must apply:

- A client supporting the Operation and Maintenance (O&M) network protocol implemented by the ME, for example a Secure Shell (SSH) version 2 client, is available. Terminal type VT100 is supported. For information on terminal types, see Section 4.1 Terminal Types on page 121.
- The user has IP access to the O&M address of the ME.



- The user has a valid O&M user account.
- Suitable firewall settings enable access to the ECLI port.

1.4 ECLI Version

The CLI version can be displayed by command **version**, see Section 3.12 Display CLI Version on page 113.

The CLI version is described using three sequence numbers, `<major>.<minor>.<revision>`, where:

- `<major>` is incremented when a significant and non-backward compatible change is made, for example, when a command is deleted, or when the syntax or behavior of a command is changed.
- `<minor>` is incremented when a significant but backward compatible change is made, for example, when a new command is added or when new options are added to an existing command.
- `<revision>` is incremented when a minor change not affecting the behavior is made, for example, when a fault that made a documented function unusable is corrected.

The version number informs CLI users, including scripts and programs, about what behavior to expect from the CLI commands. Usability enhancing features, like auto-completion and help, are not considered in the version number.

Commands labeled as optional in this document are not considered in the version number.

This document describes CLI version 1.5.0.

1.4.1 Changes between CLI Versions 1.4.0 and 1.5.0

CLI 1.5.0 adds support for the following:

- Fetch real time counter values by using new filter options `-g` and `-m` for Performance Management (PM) groups and measured objects respectively and `-f` option to format the output.
- Re-initializing an object from any cursor location.

CLI 1.5.0 removes constraint validation during navigation and errors can be checked only by issuing a `validate` or `commit` command.



1.4.2 Changes between CLI Versions 1.3.0 and 1.4.0

CLI 1.4.0 adds support for the following:

- Show command and its variants, upon failing to retrieve information from a certain MO instance, or group of MO instances with MO SPI error, continues to display information from the following MO instances, if any.
- Comments in CLI input using comment delimiter.
- Unidirectional association with external objects.

1.4.3 Changes between CLI Versions 1.2.1 and 1.3.0

CLI 1.3.0 adds support for the following:

- Retrieving model information using the new parameters `--moc`, `--property`, and `--tree` to command `help`.
- Regular expressions in search conditions.
- Performing operations on multiple MO instances using the new pipe commands `action`, `modify`, and `no`.

CLI 1.3.0 improves **Tab** completion support for typed MO reference values.

CLI 1.3.0 removes auto-completion on the comma (`,`), equal sign (`=`), and space (**Space**) keys.

1.4.4 Changes between CLI Versions 1.2.0 and 1.2.1

CLI 1.2.1 changes the name of command `reset` to `reinit`.

1.4.5 Changes between CLI Versions 1.1.1 and 1.2.0

CLI 1.2.0 adds support for the following:

- Command `reset` to set an attribute or an MO instance back to its default state.
- Command `back` to navigate back to a previously visited CLI position.
- Recursive searching with search conditions in command `show` and `show-table`.
- Searching for an MO instance to navigate to in command `dn`.
- More escape sequences in string values.
- Passphrase strings.
- Addressing individual values in sequences of simple types using indexes.



- **Tab** completion of multiple attribute assignments on the same command line, and of MO reference values.

1.4.6 Changes between CLI Versions 1.1.0 and 1.1.1

CLI 1.1.1 adds option `-s` | `--sort` to command `show`, `show-config`, `show-mib`, and `show-table`. With this option, MO instances are sorted according to their instance names.

1.4.7 Changes between CLI Versions 1.0.0 and 1.1.0

CLI 1.1.1 adds support for the following:

- Floating point numbers.
- MO key attributes of integer and enumeration types. This does not affect the CLI syntax, but new error messages specific to these types can be returned when attempting to create an MO instance.
- Properties `canCreate` and `canDelete` on MO relations. This does not affect the CLI syntax, but new error messages specific to these properties can be returned when attempting to create or delete an MO instance.





2 CLI Concepts

This section describes the CLI concepts.

2.1 CLI Modes

The CLI provides the following modes:

- Exec mode – Displays the status of the ME. In this mode, the user enters commands to monitor the ME, display its configuration, and execute actions.
- Config mode – Used to change the ME configuration. In this mode, the user starts a configuration transaction to the MIB, enter commands to change the ME configuration, and commit the changes.



Attention!

Risk of data loss or data corruption.

Unless stated otherwise, only execute actions in Exec mode to minimize the risk to misconfigure the system.

As shown in Figure 1, when the user initiates a CLI session, the user always enter Exec mode, which is the default mode.

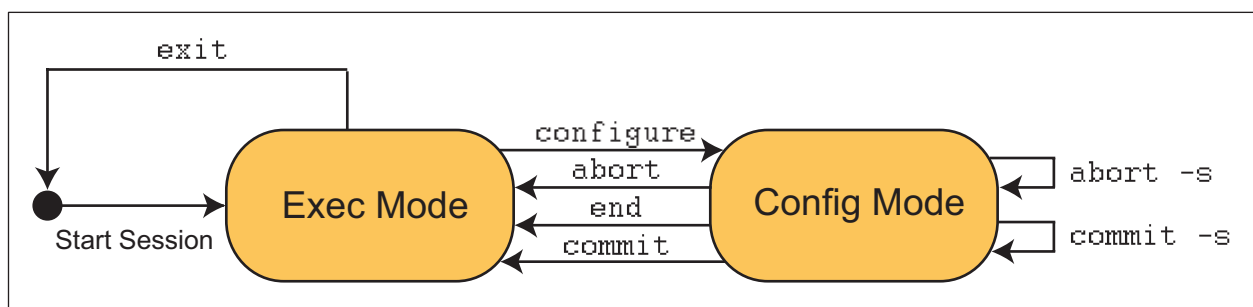


Figure 1 CLI Modes

Depending on the CLI mode, different CLI functions and set of CLI commands are supported with different parameters.



2.2 CLI Session

This section describes a CLI session. A CLI session can be started over SSH and TLS.

2.2.1 Start a CLI Session over SSH

How to start a CLI session depends both on the O&M network protocol implemented by the ME and on the client. This section describes two common cases:

- With a standard SSH client towards an ME that provides a dedicated CLI SSH port on the O&M address.
- With a standard SSH client towards an ME that provides an SSH port that is shared with between services.

2.2.1.1 Log On to CLI Session over SSH

To start a CLI session:

1. Use a terminal and start the SSH session.

Example of logon with OpenSSH client:

```
ssh <user>@<target_host> -p 22
```

The options are as follows:

- <user> – Username.
- <target_host> – The OAM virtual IP address of the ME.
- -p (port number) – TCP port 22 is default.

Root user access is denied.

2. Wait for the session to start.

2.2.1.2 Log On to CLI Session over SSH (Deprecated)

To start a CLI session:

1. Use a terminal and start the SSH session.

Example of logon with OpenSSH client:

```
ssh <user>@<target_host> -p 22 -t -s cli
```



The options are as follows:

- **<user>** – Username.
- **<target_host>** – The OAM virtual IP address of the ME.
- **-p** (port number) – TCP port 22 is default.
- **-t** (force pseudo-tty allocation).
- **-s** (subsystem) – Use **cli**.

Root user access is denied.

2. Wait for the session to start.

2.2.1.3 Successful Logon

When logon is successful, a startup message can be displayed (if it is configured), the session starts in Exec mode, and the prompt is displayed.

2.2.1.4 Unsuccessful Logon

Logon can fail because of the reasons specified in Table 2.

Table 2 Logon Error Messages

Error Message	Recommended Action
SSH session timeout	Check the IP address and port accessibility.
Invalid Credentials	Get a valid pair of user account and password from the System Administrator.
Connection to COM failed (<socket_error>)	Check if the CLI service is running.
subsystem request failed on channel 0	Check the SSH server state and verify that its configuration is valid.

2.2.2 Start a CLI Session over TLS

2.2.2.1 Log On to CLI Session over TLS

Start a TLS connection

Note: To support auto completion and control signals, set the terminal to raw mode.

Example of logon with OpenSSL **s_client**:

1. Set the terminal to raw mode in bash:



```
stty raw -echo
```

2. Launch openssl s_client

```
openssl s_client -connect <host>:<port> -quiet -tls1 -bugs  
-cert <nodecert> -key <nodekey> -CAfile <cacert>
```

The options are as follows:

- **<host>** – O&M IP address or hostname of the ME.
- **<port>** – Transmission Control Protocol (TCP) port 6522 is default.
- **<nodecert>** – Certificate file to use, PEM format is assumed.
- **<nodekey>** – The private key to use. If not specified, the certificate file is used.
- **<cacert>** – PEM format file of CAS.

Note: The client certificate contains the user identity in field `subjectAltName`.

3. Remove raw terminal settings:

```
stty sane
```

2.2.2.2 Successful Logon

When logon is successful, a startup message can be displayed, the session starts in Exec mode, and the prompt is displayed.

2.2.2.3 Unsuccessful Logon

TLS client-specific error messages are displayed. For more information, refer to respective client documents (example: OpenSSL s_client).

2.2.3 Parallel Sessions

The maximal number of parallel sessions is 256.

2.2.4 Session Inactivity Timer

When the predefined time has passed without activity in the CLI, the system automatically aborts the ongoing transaction and closes the session without any warning. Activity in a CLI session means any operation resulting in data exchange between the terminal and the server.

The default inactivity timer value is 120 seconds.



2.2.5 CLI Session End

CLI session is closed either as a result of command **exit** or when the session inactivity timer expires.

2.3 CLI Transaction

Configuration changes are applied in a transaction in such a way that all or none of the operations are executed.

If an action executed in Exec mode changes the configuration, the change is committed automatically when the action is executed. In Config mode, all configuration changes, whether from configuration commands or from actions, are committed with command **commit [-s|--stay]**.

2.3.1 Start

The CLI session starts in Exec mode and can be changed to Config mode by command **configure**.

If the CLI mode is changed to Config mode, this initiates a new configuration transaction to the MIB.

2.3.2 Commit

Command **commit [-s|--stay]** validates the transaction and, on success, commits the configuration changes. The CLI position is unchanged, unless the MO in the CLI position no longer exists; in this case the CLI position moves to the closest instantiated parent MO location.

Command **commit** without any argument performs a commit and the CLI mode changes to Exec mode.

```
(config-ManagedElement=NODE06ST)>userLabel="Stockholm, Building 25"
(config-ManagedElement=NODE06ST)>commit
```

Example 1 Command commit

Command **commit -s** or **commit --stay** performs a commit, the CLI remains in Config mode and starts a new transaction instead of returning to Exec mode.

```
(config-ManagedElement=NODE06ST)>userLabel="Stockholm, Building 25"
(config-ManagedElement=NODE06ST)>commit -s
```

Example 2 Command commit -s

If the command is completed without errors, nothing is displayed by the system.



From the CLI perspective, a successful **commit** command consists of following three steps:

1. Validates the data in the transaction against model restrictions, such as multiplicity and cardinality.
2. Requests validation of the transaction from the middleware.
3. Requests transaction **commit** from the middleware.

If any of the three steps fail, an error message is displayed and the command does not proceed to the next step.

Validation errors related to multiplicity and cardinality from all the MOs are thrown when **validate** or **commit** command is executed, as shown in Example 3.

Note: It is recommended to use command **validate** before **commit** to verify that the entered changes are valid.

```
(config-ManagedElement=NODE06ST)>TestRootMoc=1
(config-TestRootMoc=1)>..,TestRootMoc=2
(config-TestRootMoc=2)>commit
ERROR: Unable to commit incomplete object (ManagedElement=NODE06ST,TestRootMoc=1)
Minimum multiplicity of 1 is violated for attribute (mandatoryBoolMultivalueAttr)
Minimum multiplicity of 2 is violated for attribute (stringMultivalueAttr)
Current cardinality of 0 for class-instance of TestRootChildMoc is < 1 (lower-limit) for
ManagedElement=NODE06ST,TestRootMoc=1
ERROR: Unable to commit incomplete object (ManagedElement=NODE06ST,TestRootMoc=2)
Minimum multiplicity of 1 is violated for attribute (mandatoryBoolMultivalueAttr)
Minimum multiplicity of 2 is violated for attribute (stringMultivalueAttr)
Current cardinality of 0 for class-instance of TestRootChildMoc is < 1 (lower-limit) for
ManagedElement=NODE06ST,TestRootMoc=2
(config-TestRootMoc=2)>
```

Example 3 Commit Command

2.3.2.1 Model Validation Phase Error Messages

When command **commit** fails during the model validation, model restrictions have been violated by the data in the current transaction. The error messages are described in Table 3.

When the model validation is unsuccessful, the transaction is still valid and its state is the same as before command **commit** was executed.

Table 3 Model Validation Phase Error Messages

Error Message	Description
ERROR: Unable to commit incomplete object (<path>) <error_specific_information_why_the_validation_failed>	The error-specific part gives detailed information about why the validation failed.



2.3.2.2 Middleware Validation Phase Error Messages

Command **commit** can fail during the middleware validation phase. The error messages are described in Table 4.

Table 4 Middleware Validation Phase Error Messages

Error Message	Description
ERROR: Transaction not committed due to validation errors Transaction validation failed! <error_specific_information_why_the_validation_failed,_when_available>	The transaction is still valid and its state is the same as before the command commit was executed.
ERROR: Transaction validation failed with error code: <error_code>	The validation failed unexpectedly in the middleware. The error code returned from the middleware can be interpreted with help of Section 3.2.1 Error Codes on page 47. The transaction is still valid and its state is the same as before command commit was executed, unless the error code definition states otherwise.

2.3.2.3 Commit Phase Error Message

When command **commit** fails during the commit phase, an error message is displayed, which is described in Table 5.

Table 5 Commit Phase Error Message

Error Message	Description
ERROR: Transaction commit failed and uncommitted changes have been lost	The transaction cannot be recovered. The transaction and all the changes associated with it are lost.

The following warning is displayed to notify that the prompt is changed because of cursor points to an uninstantiated MO:

Invalid location. Cursor points to uninstantiated MO

This can occur if command **commit** fails and the MO that the cursor points to was created in the transaction that was lost.

2.3.3 Abort

Command **abort** [-s|--stay] discards the changes in the transaction and terminates the transaction.

Command **abort** without any argument performs an abort and, on success, returns to Exec mode.



```
(config-ManagedElement=NODE06ST)>userLabel="Sto  
ckholm, Building 25"  
(config-ManagedElement=NODE06ST)>abort
```

Example 4 Command abort

Command **abort** -s or **abort --stay** performs an abort and starts a new transaction. The CLI position is unchanged, unless the MO in the CLI position no longer exists; in this case the CLI position moves to the closest instantiated parent MO location.

```
(config-ManagedElement=NODE06ST)>userLabel="Sto  
ckholm, Building 25"  
(config-ManagedElement=NODE06ST)>abort -s
```

Example 5 Command abort -s

If the abort fails, this results in an error message and the CLI mode is unchanged.

2.3.4

End

Command **end** returns from Config mode to Exec mode when there are no changes in the configuration transaction. The CLI position is unchanged, unless the MO in the CLI position no longer exists; in this case the CLI position moves to the closest instantiated parent MO location.

Use command **abort** or **commit** to return to the Exec mode after entering configuration changes.

```
(config-ManagedElement=NODE06ST)>end
```

Example 6 Command end

If transactional changes have been made, the command returns error message Configuration changes have been made in the current transaction, use 'abort' or 'commit' to leave config mode.

2.3.5

Time-Out

At transaction inactivity time-out, the following events are triggered:

- Transaction abort resulting in deletion of transaction content and MO locks.
- State of the transaction is changed from active to timed out.
- Start of transaction cleanup timer triggering transaction ID and state deletion after a system-defined period, which is 3600 seconds by default.



The first operations entered in a timed out, but not deleted, transaction is replied with the following transaction time-out error message:

```
ERROR: Transaction broken, possibly because of a timeout.
Uncommitted changes have been lost
```

If the session inactivity timer occurs before transaction time-out, the session is closed and the transaction is immediately aborted.

2.3.6 Exit

Command **exit** exits the CLI session. This command is valid only in the Exec mode.

2.4 CLI Position

The CLI position is a Distinguished Name (DN) of an MO instance identifying a position in the MO tree. The CLI position can be changed by navigation commands, such as directory change commands in a directory tree of a file system.

2.4.1 Distinguished Name

The DN identifies an MO instance with comma-separated sequence of `<class_name>=<key_value>` name value pairs.

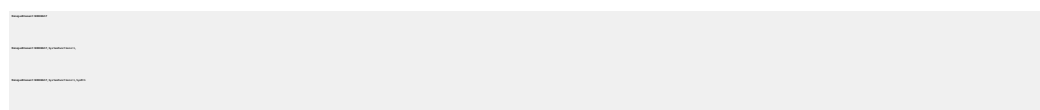
Example 7 Distinguished Name

2.4.2 Local Distinguished Name

The Local Distinguished Name (LDN) is a DN starting at LDN root and provides address of MO instance through its sequence of parent MOs. The LDN root is `ManagedElement=<managed_element_id>`.

The MOs are organized in a hierarchical structure. Each MO instance is uniquely identified in the node by its LDN. The highest MO in a node, the root MO, is `ManagedElement` and represents the whole node.

When an MO is located further down in the MO tree, the LDN must contain the MO classes identifying all parents of that MO, in a sequence going from the root MO down to the MO in question.



Example 8 Local Distinguished Name



In this example, `ManagedElement=NODE06ST` has child `SystemFunctions=1`, which has child `SysM=1`, which has child `Snmp=1` representing `Snmp`. The LDN of the lowest MO (`Snmp=1`) contains the address of all successive parents of that MO all the way up to `ManagedElement=NODE06ST`.

2.4.3 Relative Distinguished Name

The Relative Distinguished Name (RDN) is the address of an MO instance in relation to its parent MO. It is a `<name_value>` pair DN.

For example, `SysM=1` is the RDN for LDN `ManagedElement=NODE06ST, SystemFunctions=1, SysM=1`.

2.4.4 CLI Path

An CLI path, denoted `<path>`, can consist of either an LDN or an RDN.

2.4.5 Valid Positions in Tree

Considering the LDN `ManagedElement=NODE06ST, SystemFunctions=1, SysM=1`, the valid positions in the tree are described in Table 6.

Table 6 Valid Positions in Tree

Position in Tree	Description
Root position	The root of the MO tree containing <code>ManagedElement</code> . Indicated in the DN and <code><path></code> field of the CLI prompt as an empty string.
Position at <code>ManagedElement=NODE06ST</code>	The position at <code>ManagedElement</code> . Indicated in the DN and RDN field of the CLI prompt as <code>ManagedElement=NODE06ST</code> .
Position at <code>SystemFunctions=1</code>	The position at <code>SystemFunctions</code> . Indicated in the RDN field of the CLI prompt as <code>SystemFunctions=1</code> . Indicated in the DN field of the CLI prompt as <code>ManagedElement=NODE06ST, SystemFunctions=1</code> .
Position at <code>SysM=1</code>	The position at <code>SysM</code> . Indicated in the RDN field of the CLI prompt as <code>SysM=1</code> . Indicated in the DN field of the CLI prompt as <code>ManagedElement=NODE06ST, SystemFunctions=1, SysM=1</code> .



2.4.6 Interpretation of CLI Commands

CLI commands are interpreted in the context of the CLI position. That is, attribute names, action names, and RDNs are prepended by the DN indicating the CLI position. For example, the following operations show the same attribute from different CLI positions:

```
(config)>show ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,userLabel
```

```
(config-ManagedElement=NODE06ST)>show SystemFunctions=1,SysM=1,userLabel
```

```
(config-SystemFunctions=1)>show SysM=1,userLabel
```

```
(config-SysM=1)>show userLabel
```

2.5 CLI Prompt

The CLI prompt provides information on the status and context of the CLI session as follows:

- In Exec mode, the default prompt is >. The prompt is changed in Exec mode, based on the RDN value during navigation, to (RDN)>.
- In Config mode, the default prompt is (config)>. The prompt is changed in Config mode, based on the RDN value during creation or navigation, to (config-RDN)>.

After session start, command **prompt** changes the prompt configuration in any of the CLI modes for the lifetime of the session to contain any elements shown in Table 7.

Table 7 CLI Prompt Configuration Elements

Element	Description
\$default	The \$default value is equivalent to \$mode-\$rdn in Config mode and an empty string in Exec mode.
\$dn	LDN indicating the CLI position relative to the root position.
\$mode	CLI mode, that is, config or exec.
\$hostname	The name of the host the CLI service runs on.
\$nodename	The key value of the ManagedElement MO.
\$rdn	RDN indicating the CLI position relative to the parent MO.



Table 7 CLI Prompt Configuration Elements

Element	Description
\$user	User account name.
<Value: Any user-defined string>	<p>Value can be any user-defined string. It can also contain special characters in escaped hexadecimal format.</p> <p>Supported characters and escape sequences are \", \', \t, \n, \b, \f, \r, and \v.</p> <p>Special characters with escaped hexadecimal form are also supported, in form \x, followed by the ASCII code in hexadecimal in two digits. For example, the hash character # must be specified as \x23 and is displayed in the CLI prompt as #.</p>

If the CLI prompt is longer than the terminal width, it continues on the next line.

2.6 Context-Sensitive Help

The CLI provides online and context-sensitive help. It enables the user to access information and learn about the commands, the MIB, and the MOM without relying on the documentation library.

By pressing the ? key, the user can request context-sensitive help on the CLI operations and MOM elements (MOCs, attributes, and actions) that are available for the following:

- CLI mode
- CLI position
- CLI command

After the help text, a new prompt without the ? character is displayed. If no help text is available, only a new prompt is displayed.

Note: The CLI supports only US-ASCII characters. If any other character is to be displayed in help text, it is displayed as a question mark (?) in the CLI terminal.

Verbose Help is provided if the context uniquely identifies a single MOM element, else only Brief Help on all available elements is listed, as a summary.

Verbose Help is provided when pressing the ? key on a single element (CLI command or MOM element).

No help is provided when pressing the ? key on partial command (if context does not find any hit for MOM elements).



CLI help is not available for the position in key values of the CLI path. For example, no CLI help is provided in the following cases:

- On an attribute in command `show-config <path>`, `<attribute_name>` if the attribute is defined as read-only or restricted.
- On a MOC in command `show-config [<path>]`, `<class_name>` if the class is defined as system-created and has no MO instance created.
- On a MOC in command `show [<path>]`, `<class_name>` if the class is defined as not system-created and has no MO instance created.

Note: Command help can also be used to retrieve help on both CLI commands and the model. For more information, see Section 3.14 Help Command on page 118.

2.6.1 Brief Help

A description of the Brief Help content and format is provided in Table 8.

Table 8 Printout Syntax for Brief Help

CLI Element	Printout Syntax
Action	<code><action_name>()<spaces><truncated_action_description></code>
Action parameter	<code><parameter_name><spaces><truncated_parameter_description></code>
Class	<code>+<class_name><spaces><truncated_class_description></code>
CLI operation parameter	<code><parameter_name><spaces><parameter_description></code>
Single-valued attribute, struct member	<code><attribute_name><spaces><truncated_attribute_description></code>
Struct (multi-valued attribute)	<code><attribute_name>[]<spaces><truncated_attribute_description></code>

The class, attribute, and action descriptions are displayed in a truncated form (first sentence only). Truncation is not indicated in the printout. The text-formatting characters (tab, new line) are deleted from the description printouts.

For example, `(config-ManagedElement=NODE06ST)>?`, brings Brief Help information about the `ManagedElement` MOC attributes and the child MOCs, as shown in Example 9 and Example 10.



```
(config-ManagedElement=NODE06ST)>?
dateTimeOffset      Difference between the value of the localDateTime attribute and UTC
                    (Coordinated Universal Time).
dnPrefix            It provides naming context allowing the managed objects to be partitioned
                    into logical domains.
localDateTime       This is the local date and time for the Managed Element.
managedElementId    Contains the identity of a release of the product type being managed.
managedElementType  The type of product being managed.
networkManagedElementId Replaces the value component of the RDN in the COM Northbound Interface.
productIdentity     Contains product information for the Managed Element and its Managed
                    Function(s).
release            The release of the type of product specified by the attribute
                    managedElementType.
siteLocation        A freetext attribute describing the geographic location of a Managed
                    Element.
timeZone            This is the timeZone that the Managed Element resides in.
userLabel           A freetext string for additional information to assist Managed Element
                    identification.
+SystemFunctions    This model has a structural purpose to group the management of the system
                    functions of the Managed Element
+Transport          This is a container for common transport functions used within the Managed
                    Element.
[...]
```

Example 9 Brief Help

```
(config-MOex1=1)>addNumbers ?
```

Example 10 Brief Help on Action Parameters

The following indicators are used to indicate what syntax is used to interact with the data type:

- () action or command that can be executed
- + MO instance or struct that can be navigated to

```
(config-ManagedElement=NODE06ST)>?
```

Example 11 Indicators in Help Text

2.6.2 Verbose Help

A description on the Verbose Help content and format is provided in Table 9.

Table 9 Printout Syntax for Verbose Help

Item	Printout Syntax
Action	<action_name>() [Return Type] <action_description>
Action parameter	<parameter_name><parameter_type> [passphrase] [optional/default=<default_value>] <parameter_description>



Table 9 Printout Syntax for Verbose Help

Item	Printout Syntax
Class	<class_name> MO type [singleton][optional] <class_description> ⁽¹⁾
CLI operation	<CLI_operation_name>() Command <operation_description>
CLI operation parameter	<parameter_name> <parameter_description>
Single-valued attribute, struct member	<attribute_name> <attribute_type> [passphrase] [optional/read only/default=<default_value>/exclusive] <attribute_description> ⁽²⁾ <attribute_description> ⁽²⁾

(1) The [singleton] class specifier is present if exactly one instance of this class can exist as a child MO of its actual parent. The [optional] class specifier is present if zero is the minimal number of instances of this class as a child MO of its actual parent.

(2) The [exclusive] specifier can only be present for struct members, and means that struct property isExclusive is set. See Section 3.3.1 Display Single-Valued Attribute on page 49.

The descriptions of class, attribute, and action are displayed in a complete form without truncating the text. The text formatting characters (tab, new line) are kept.

```
>show ManagedElement=NODE06ST, SystemFunctions ?
SystemFunctions MO Type [singleton] [optional]
This model has a structural purpose to group the management of the system functions
of the Managed Element.
```

Example 12 Verbose Help on MOC

```
(config)>ManagedElement=NODE06ST, userLabel ?
userLabel String [optional]
A freetext string for additional information to assist Managed Element identification.
```

Example 13 Verbose Help on Attribute of Type String

```
(config-ManagedElement=NODE06ST)>siteLocation ?
siteLocation String [optional]
A freetext attribute describing the geographic location of a Managed Element.
```

Example 14 Verbose Help on String Attribute

If help is triggered directly after (without space) the CLI operation or action name, Verbose Help is provided on the operation, as shown in Example 15.

```
(config-ManagedElement=NODE06ST)>show?
show Command
Display information
```

Example 15 Verbose Help on CLI Operation

If help is triggered separated by space from the command or action name, Verbose Help is provided on the operation parameter, as shown in Example 16 through Example 20.



```
(config)>show ?
--moc          Option to select a specific Child MOC under the current DN
--recursive    Display all information
--sort         Sort the MO instances in numerical/alphabetical order
--verbose      Display verbose information
-m            Option to select a specific Child MOC under the current DN
-r            Display all information
-s            Sort the MO instances in numerical/alphabetical order
-v            Display verbose information
+ManagedElement The top-level class in the Common Information Model
               is Managed Element root Managed Object Class.
```

Example 16 Verbose Help on CLI Command

```
(config-M0ex2=1)>concatString_defValues --str1 ?
--str1          String [optional/default=com]
String one
```

Example 17 Verbose Help on Action Parameters

```
(config)>show-mib ?
--sort          Sort the MO instances in numerical/alphabetical order
--verbose       Display full path of MO instance information
-s             Sort the MO instances in numerical/alphabetical order
-v            Display full path of MO instance information
+ManagedElement The top-level class in the Common Information Model
               is Managed Element root Managed Object Class.
```

Example 18 Verbose Help on Command show-mib

```
(config)>show -v ?
--recursive    Display all information
--sort         Sort the MO instances in numerical/alphabetical order
-r            Display all information
-s            Sort the MO instances in numerical/alphabetical order
+ManagedElement The top-level class in the Common Information Model
               is Managed Element root Managed Object Class.
```

Example 19 Verbose Help on Command show -v

```
(config)>show-table ?
--moc          Option to select a specific Child MOC under the current DN
--recursive    Display all information
-m            Option to select a specific Child MOC under the current DN
-r            Display all information
+ManagedElement The top-level class in the Common Information Model
               is Managed Element root Managed Object Class.
```

Example 20 Verbose Help on Command show-table

```
(config)>show | filter ?
--after        Print number of lines of trailing context after matching lines
--before       Print number of lines of leading context before matching lines
--ignore       Ignore case distinctions in both the pattern and the input
--invert       Invert the sense of matching, to select non-matching lines
-A            Print number of lines of trailing context after matching lines
-B            Print number of lines of leading context before matching lines
-i            Ignore case distinctions in both the pattern and the input
-v            Invert the sense of matching, to select non-matching lines
<pattern>     The text used for pattern matching
```

Example 21 Verbose Help on Filter



2.7 Auto-Completion

By pressing the **Tab** key at any position of the CLI command, completion can be requested on all possible continuations of the CLI command and MOM-defined elements the user is authorized to.

Depending on the entered command, the response can be as shown in Table 10.

Table 10 Responses at Auto-Completion

Scenario	Response
Multiple valid continuations exist	<p>All valid continuations of the CLI command are displayed in Completion Possibility List with the following content and in the following order:</p> <ul style="list-style-type: none"> • CLI command name, for example, show or history. • CLI command parameter name, for example, -v --verbose for command show. • MOC name. • Attribute name and struct member name. • Action name. • Action parameter name. • Value of key attribute of existing MOs. • <new> indicates in Config mode that a key attribute value of a new MO can be specified. • <value> indicates in Config mode that a new value for an attribute (not readOnly) can be specified. • " indicates in Config mode that a string attribute (not readOnly) value can be specified in quotation marks. • [indicates in Config mode that a sequence attribute (not readOnly) values can be specified within square brackets. • Comma (,) indicates that the MO identified by the RDN can have an attribute, a child MO, or an action. • <cr> indicates that the entered CLI command is valid by itself, and command execution can be requested by pressing the CR key or the Enter key. • <space> indicates that a space can be entered after an action parameter name or the value.



Table 10 Responses at Auto-Completion

Scenario	Response
Exactly one valid continuation (unique match) exists	Auto-completion automatically adds them, including the following elements: <ul style="list-style-type: none">• Equal sign (=), comma (,), and space (" ") characters.• MOC and action names.• Attribute names.• Action parameter names.• Value of single-valued attributes (enumeration, string, integer, and boolean).• Struct member names.• Struct member names in sequence of struct if the struct has the key member.• Enum struct member values.
The entered CLI command is invalid	Auto-completion or a completion possibility list is not provided.

Auto-completion is provided only for valid CLI commands the user is authorized to, in a context-sensitive way, for example, as follows:

- Completion is provided for DNs as parameter of command **show** if the MO exists and the user has read privilege to the MO.
- Completion is provided for attribute names as parameter of command **show** if the attribute has a value assigned and the user has read privilege to the attribute.
- Completion is provided for an attribute name in an attribute value change operation if the attribute is not defined as `readOnly` or `restricted`, and if the user has write privileges to the attribute.
- Read-only and restricted attributes are not auto-completed as parameters of command **show-config**.
- Completion is provided for parameters of an action if the parameter exists.

2.7.1 Keys Activating Auto-Completion

The keys in Table 11 activate auto-completion in the listed contexts if the entered partial name uniquely identifies the name of the command or its parameter.

Table 11 Keys Activating Auto-Completion

Key	Description
Enter (CR)	Completes operation, attribute, action names, and DNs and then executes the completed command, if there is a unique match.
Tab	Provides auto-completion or a completion possibility list, or both.

2.7.2 Examples of Completion Possibility Lists

This section provides examples of completion possibility lists.

```
(config)#show
```

Example 22 Completion Possibility List of Command show without Space

```
(config-M0ex3=1)>add_
```

Example 23 Completion Possibility List for Actions without Space

```
(config-M0ex4=1)>addNumbers --n
```

Example 24 Completion Possibility List for Action Parameters

```
(config)#show
```

Example 25 Completion Possibility List of Command show with Space



```
(config-ManagedElement=NODE06ST)>show | filter_
```

Example 26 Completion Possibility List for Filter Parameters

2.7.3 Examples of Auto-Completion

Examples of auto-completion are shown in Table 12.

Table 12 Examples of Auto-Completion

Function	Input	Result
Unique match for DN	show M<Tab>	show ManagedElement=NODE06ST
	show m<Enter>	Auto-completion and execution of show ManagedElement=NODE06ST
	In Config mode in root position: Press <Enter> with empty command line	Navigation to ManagedElement=NODE06ST
Unique match for partial DN	show ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=A<Tab>	show ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=AlarmLogs if the FileGroup=AlarmLogs and FileGroup=AlertLogs MOs exist
Unique match for operation name	In Exec mode: c<Enter>	Triggers Config operations
Multiple matches for parameter names of an action	addNumbers <Tab>	addNumbers --num1 --num2
Unique match for value of a parameter of an action	addNumbers --num1<Tab>	addNumbers --num1 <value>



Table 12 Examples of Auto-Completion

Function	Input	Result
Unique match for parameter name of an action	<code>addNumbers --num1 20 <Tab></code>	<code>addNumbers --num1 20 --num2</code>
Multiple matches for <space>, <cr>, and <value> of a parameter with value of an action	<code>addNumbers_defValues --num1 23 <Tab></code>	<code>addNumbers_defValues --num1 23 <value> <space> <cr></code>

2.7.4 Auto-Completion of MO Reference Type

Completion of MO reference values are supported when the value starts with a quote. The behavior is different depending on the type of the MO reference. A generic reference is allowed to have any MOC instance as a value, whereas a typed reference value is restricted to instances of a specific MOC.

2.7.4.1 Generic MO References

For generic references, **Tab** completion suggests both absolute MO paths starting from the root MOC, and relative paths starting from the current CLI position.

Assume a model with MO D under C, MO C under B, MO B under the root ManagedElement=NODE06ST, and MoReferenceAttribute under C. This creates the following example output:

```
(config-ManagedElement=NODE06ST,B=1,C=1)>moRefere
nceAttribute="<Tab>
"ManagedElement=NODE06ST
"D
" ..
```

Assume a model with MO C1 and C2 under B, MO B under the root ManagedElement=NODE06ST, and MoReferenceAttribute under C1. This creates the following example output:

```
(config-ManagedElement=NODE06ST,B=1)>C1=1,moRefere
nceAttribute="<Tab>
"ManagedElement=NODE06ST
"C1
"C2
" ..
```



2.7.4.2 Typed MO References

For typed references, **Tab** completion suggests LDNs of existing instances of the MOC type of the reference attribute. If no such instances exist, **Tab** completion falls back to the same behavior as for generic references.

Assume a reference attribute named `targetRef` of MOC type `Target`. `Target` can be found in two different locations, in `ManagedElement`, `ParentOne`, `Target` and `ManagedElement`, `ParentTwo`, `Target`. See Example 27 through Example 29.

```
(config-SomeMO=1)>targetRef="_  
(config-SomeMO=1)>targetRef="ManagedElement=N0DE06ST,ParentOne=1,Target=1"
```

Example 27 Exactly One Instance of Target Exists

```
(config-SomeMO=1)>targetRef="<Tab>  
  
(config-SomeMO=1)>targetRef="ManagedElement=N0DE06ST,ParentOne=1,Target=
```

Example 28 Two Instances of Target Exist, Same Parent

```
(config-SomeMO=1)>targetRef="_  
  
(config-SomeMO=1)>targetRef="ManagedElement=N0DE06ST,Parent
```

Example 29 Two Instances of Target Exist, Different Parents

The CLI falls back to the behavior of generic MO references in the following situations:

- There are no instances of the MOC type of the reference attribute.
- A partial MO path has been entered, and no instances matching this path exist.
- The search algorithm finds more than 100 instances of the MOC type. The search then aborts to avoid too long response time, and a partial MO path matching less than 100 instances must be entered for suggestions to be provided.
- A relative MO path has been entered.

2.7.4.3 External MO References

For MO references with an external Unidirectional Association, **Tab** completion suggests the `<value>` tag, in addition to the suggestions for internal associations, if there are any.

Assume a reference attribute named `targetRef` with external Unidirectional Association and internal association of MOC type `target`. This creates the following example output:



```
(config-SomeMo=1)>targetRef="<TAB>
ManagedElement=1,TargetMoc=1"
<value>
(config-CrazyThing=1)>targetRef="ManagedElement=1,TargetMoc=1"
```

2.8 Case Correction

When input contains only lower case, a unique match of auto-completion automatically triggers case correction (for example, logicalfs to LogicalFs) of one CLI command element at a time for the following types:

- MOC name
- Attribute name, struct name, and struct member attribute name
- Enumeration value, that is, enumeration member or literal name
- Action name
- CLI operation and parameter name
- Action parameter name

Case correction is not supported in following cases:

- For string attribute values.
- If input contains at least one upper case.

2.9 Escaping of Special Characters

The CLI support the US-ASCII character set. In both input and output strings, escape sequences are used to represent non-printable characters, non-US-ASCII characters, and characters with a special meaning in the CLI, as shown in Table 13.

Note: A slight difference exists between the escape sequences used in attributes, struct members, and action parameters, and those used in MO instance names.

Table 13 Escape Sequences in Strings

Escape Sequence	Description
\n	New line
\r	Carriage return
\t	Tab
\"	Quotation mark



Escape Sequence	Description
\\	Escape character
\?	Question mark ⁽¹⁾
\!	Exclamation mark ⁽¹⁾
\#	Hash ⁽¹⁾
\xNN	Any character in hexadecimal format (2 hexadecimal digits). N is any character 0–9, A–F, or a–f. The sequence \x00, which translates to the string termination character in C, is not allowed. Non-US-ASCII characters are not allowed (NN is greater than \x7F).
\NN	In MO instance names only. Any character in hexadecimal format (2 hexadecimal digits). N is any character 0–9, A–F, or a–f. This format is mandated by 3GPP TS 32.300, Naming convention for Managed Objects. This escaping format is always used when DNs are displayed.

(1) This character does not need to be escaped in quoted input strings.

```
(config-ManagedElement=NODE06ST)>userLabel=hello\nworld
(config-ManagedElement=NODE06ST)>show userLabel
```

Example 30 Set String Attribute with Special Character

```
(config-aSimpleStruct)>str1=hello\!world
(config-aSimpleStruct)>show str1
```

Example 31 Set Struct Member with Special Character

If a non-supported escape sequence is entered, the CLI displays an error message.

```
(config-ManagedElement=NODE06ST)>userLabel=temp\76value
```

Example 32 Set String Attribute with Non–Special Character

Entered escape sequences are internally converted into the characters they represent before being stored. When a string value is printed by the CLI, characters are escaped when needed. For example, a quotation character within a string is printed as \".



The exception is DNs, where an escape sequence is only converted into the character it represents when the character is allowed according to 3GPP TS 32.300. Otherwise escaping is translated to the \NN format (as mandated by 32.300).

2.10 CLI Comments

It is possible to include comments in CLI input. The comment delimiter # and the rest of the input line on which it appears is considered as comment and is discarded before the line is executed. An input line that only consists of a comment is ignored, as if it had been a blank line.

CLI command completion and context sensitive help cannot be used after the # character. Pressing the **Tab** key has no effect, and pressing the **?** key will add the character to the comment.

If a # character is escaped as described in Section 2.9 Escaping of Special Characters on page 29, or if it appears inside a quoted string, it is treated as a normal string character.

2.11 Visibility Levels

The behavior of CLI command output depends on the following visibility levels returned for MOM elements:

- `visible` – Specifies that the user has full access to all MOM elements.
- `accessible` – Specifies that the user can only display or access MOM elements by providing the full name of a MOM element.
- `not-visible` – Specifies that the user cannot display or access MOM elements.

The behavior of the visibility for MOM elements in the CLI is shown in Table 14.



Table 14 Visibility Behavior

Resulting Entity Visibility Level	visible	accessible	not-visible
Normal show Output	Displayed	Displayed only in the following cases: <ul style="list-style-type: none">• When user is located in accessible element• When user runs <code>show -r</code> from accessible parent element• When user runs <code>show</code> using DN to location Not displayed when running <code>show -r</code> from visible parent element.	Not displayed. If <code>show</code> is run on DN, error message Element not visible is returned
<code>show -v --verbose [-r --recursive]</code>	Displayed	Displayed. Status value is added, for example, <code><deprecated></code>	Not displayed. If <code>show</code> is run on DN, error message Element not visible is returned
<code>show-config [-v --verbose]</code>	Displayed	Displayed	Not displayed
Auto-Completion	Displayed	Supported only in following cases: <ul style="list-style-type: none">• When user is located in accessible element• When user is located in accessible parent element Not supported when user is located in visible parent element	Not supported
Help on Empty Command Line	Displayed	Supported only in following cases: <ul style="list-style-type: none">• When user is located in accessible element• When user is located in accessible parent element Not supported when user is located in visible parent element	Not supported
Help on Element Name	Displayed	Supported	Not supported
Navigation	Supported	Supported	Not supported



The default visibility level for all MOM elements is `visible`. That is, `obsolete`, `preliminary`, and `deprecated` elements are set to `visible` in visibility configuration file for backward compatibility.

Note: The examples are executed with `deprecated` element set to `accessible`. The `obsolete` and `preliminary` elements are set to `not-visible`.

In the examples, consider the visibility level for the following MOC instances:

- `YCurrentThing` MOC is `visible`
- `YDeprecatedThing` MOC is `accessible`
- `YObsoleteThing` MOC is `not visible`

2.11.1 Auto-Completion

Auto-completion for accessible element `YDeprecatedThing` is not supported, as the current location is visible element `YCurrentThing`, as shown in Example 33.

```
(config-YCurrentThing=1)>YDepre_
```

Example 33 Auto-Completion of Accessible Element

Auto-completion for accessible element `YSampleDeprecatedThing` is supported, as the current location is accessible element `YDeprecatedThing`, as shown in Example 34.

```
(config-YDeprecatedThing=1)>YSample_
```

Example 34 Auto-Completion of Accessible Element

2.11.2 Display Information

When command `show` is executed from the visible current location, only visible elements, such as `YCurrentThing`, are displayed. as shown in Example 35.

```
(config-YCurrentThing=1)>show
```

Example 35 Command show

When command `show -v|--verbose` is executed from the visible current location, all visible elements, such as `YCurrentThing`, and accessible elements, such as `YDeprecatedThing`, are displayed. When the life cycle status of an element is anything else than the current one, the status is included as a tag in verbose mode, as shown in Example 36.



1. **What is the main purpose of the study?**
 2. **What are the research objectives?**
 3. **What is the research methodology?**
 4. **What are the results of the study?**
 5. **What are the conclusions of the study?**
 6. **What are the limitations of the study?**
 7. **What are the implications of the study?**
 8. **What are the future research directions?**

When command **show-config** is executed from the visible current location, all visible elements, such as YCurrentThing, and accessible elements, such as YDeprecatedThing, are displayed, as shown in Example 37.

[illegible]

Context-Sensitive Help



When the life cycle status of an element is anything else than the current one, the status is included as a tag in the context-sensitive help information, as shown in Example 38.

```
(config-YCurrentThing=1)>YDeprecatedThing?
```

Example 38 Context-Sensitive Help for Accessible Element

For not-visible elements, no context-sensitive help is supported.

2.12 Automatic Correction of ManagedElement ID

The `ManagedElement` ID typically reflects the node name, that is, different nodes in a network can have different names.

To facilitate creation of generic CLI scripts and other tools to be used for several nodes, the CLI accepts any 3GPP® compliant `ManagedElement` ID as part of a DN. The CLI can automatically correct this to the actual/correct ID.

This applies for all commands and command parameters containing a DN.

2.13 Limitations

The CLI has the following limitations:

- CLI session properties (prompt, width, length, script mode) changed in an CLI session are not stored persistently and are lost after session end.
- The maximum line length supported by the CLI is 2048 characters.





3 CLI Commands

An overview of the CLI commands is provided in Table 15.

Table 15 CLI Commands

Description	Commands
Commands to browse information	show, show-config, show-counters, show-dn, show-mib, show-table
Navigation commands to navigate in the MIB without changing it	dn, top, up, back
Commands to modify information in the MIB; these require a transaction and are therefore only available in Config mode	insert, no, reinit
Transaction commands to interact with a transaction	abort, commit, configure, validate, end
Commands to modify and request information about the CLI behavior	help, history, length, prompt, scriptmode, version, width
Pipe command	filter, format, modify, action, no
Other commands	?, exit, passwd, <RDN>

3.1 Summary of CLI Commands

A summary of the CLI commands is provided in Table 16.

Table 16 Summary of CLI Commands

Command	Mode	Description
?	Exec Config	Displays context-sensitive help on CLI operations and MOM elements. For details, see Section 2.6 Context-Sensitive Help on page 18.
abort [-s --stay]	Config	Discards changes in the transaction and terminates the transaction. For details, see Section 2.3.3 Abort on page 13.



Table 16 Summary of CLI Commands

Command	Mode	Description
back [-h --history]	Exec Config	Navigates back to the previous position in the MO tree with the following option: <ul style="list-style-type: none">• -h --history – Lists the 10 previous positions in the MO tree without changing the current position. The root position in the navigation history is not stored in the navigation history, as command top can be used instead.
commit [-s --stay]	Config	Validates the transaction and, on success, commits the configuration changes. For details, see Section 2.3.2 Commit on page 11.
configure	Exec	Changes the CLI mode from Exec mode to Config mode.
dn <LDN> dn -m --moc <moc_name> [-c --condition <condition>]	Exec Config	Navigates to any location in the MOM with the following options: <ul style="list-style-type: none">• -c --condition – Criteria for the MO to navigate to. This can be used if there is more than one instance of the specified MOC.• -m --moc – The MOC to navigate to. For details, see Section 3.3.9.1 Parameters to Filter MO Information on page 63.
end	Config	Changes the CLI mode to Exec mode, if there are no changes in the transaction. For details, see Section 2.3.4 End on page 14.
exit	Exec	Exits the CLI session. For details, see Section 2.2.5 CLI Session End on page 11.



Table 16 Summary of CLI Commands

<pre>(<command> <action>) filter [-i --ignore] [-v --invert] [{-A --after} <value>] [{-B --before} <value>] <pattern></pre>	Exec Config	<p>Displays the lines matching the specified pattern with the following options:</p> <ul style="list-style-type: none"> • -i --ignore – Ignores case distinctions in both the pattern and the input. • -v --invert – Inverts the sense of matching, to select non-matching lines. • -A --after – Displays number of lines of trailing context after matching lines. • -B --before – Displays number of lines of leading context before matching lines. • <value> – Number of lines to be displayed after leading/trailing context. • <pattern> – A regular expression used for filtering the output of a command/action. It can have alphanumeric characters and special characters supported by POSIX® regular expressions. <p>For details, see Section 3.13.1 Filter Command on page 113.</p>
<pre>help [-m --moc] <moc_name> [-p --property <property_name> >[,<property_name>]...]]</pre>	Exec Config	<p>Without parameter, provides help on the current CLI mode and the available commands in this mode. With parameters, provides help on the information model. For details, see Section 3.14 Help Command on page 118.</p>
<pre>help -t -tree [-m --moc <moc_name>]</pre>		
<pre>history [-s --size] [<number>]</pre>	Exec Config	<p>Without parameter, displays the command history of the CLI session in chronological order in format <sequence_number> <date> <time> <command> with the following options:</p> <ul style="list-style-type: none"> • -s --size – Specify the size. • <number> – Specify the number of lines to be displayed. Default is 100. <p>This command is limited to the 100 latest commands. If <number> is greater than 100, the text The command history is limited to the 100 latest commands is displayed at the end of command output.</p>



Table 16 Summary of CLI Commands

<code>insert [<path>,] <attribute_name>'['<existing_sequence_element>'<new_sequence_element>' (1)</code>	Config	Inserts simple type elements to a sequence or inserts an struct element to a sequence of keyless structs. For details, see Section 3.5.3.3 Insert Simple Type Elements to Sequence on page 83, Section 3.5.3.4 Insert Simple Type Element in Sequence Using Index on page 84, and Section 3.5.3.13 Insert Struct Element to a Sequence of Keyless Structs on page 92.
<code>insert [<path>,] <attribute_name>'['@<index>']'<new_value>' (2)</code>		
<code>insert [<path>,] <noKeyStructSequence>'['@<position>']' (3)</code>		
<code>length [<Length>]</code>	Exec Config	Without parameter, displays the number of CLI output rows until --More-- is printed and print is suspended. The printout is continued by pressing the Space key or Enter key, and discarded by pressing the Q key. Parameter Length is a number in the range 0–2147483647, except 1. Default is zero, indicating no output break.
<code>no <path></code>	Config	Deletes an MO, or an attribute value, or an element in a sequence, or all elements in a sequence, or an element at position in a sequence of keyless structs, or all struct elements in a sequence. For details, see Section 3.8 Delete MO on page 107, Section 3.5.2 Delete Value of Single-Valued Attribute on page 82, Section 3.5.3.7 Delete Named Element from Sequence on page 87, Section 3.5.3.8 Delete Element in Sequence Using Index on page 87, Section 3.5.3.9 Delete All Elements from Sequence on page 88, and Section 3.5.3.12 Delete Keyless Struct Sequence on page 91.
<code>no <nillable_attribute_name></code>		
<code>no <attribute_name>=<sequence_element_value></code>		
<code>no [<path>,] <attribute_name></code>		
<code>no [<path>,] <attribute_name>'['@<index>']' (2)</code>		
<code>no [<path> <noKeyStructSequence>'['@<position>']' (3)</code>		
<code>no <sequence_name></code>		
<code>no <noKeyStructSequence></code>		
<code>passwd</code>	Exec Config	Changes the CLI user password in root position. This is an optional command. It is available only if the ME supports changing user password through the CLI. For details, see Section 3.11 Change Password Command on page 112.
<code>[<path>,] [.,] <action_name> [--<action_parameter_name> <action_parameter_value>] ...</code>	Exec Config	Requests action execution. For details, see Section 3.9.1 Action Request on page 108.
<code>[<path>,] <attribute_name>=<attribute_value></code>	Config	Assigns value to an attribute. For details, see Section 3.5.1 Change Single-Valued Attribute on page 80.



Table 16 Summary of CLI Commands

<code>[<path>,<attribute_name>]'<existing_sequence_element>']'=<new_sequence_element_value>⁽¹⁾</code>	Config	Changes or adds a sequence element. For details, see Section 3.5.3.5 Change Sequence Element on page 85 and Section 3.5.3.6 Change or Add Sequence Element Using Index on page 85.
<code>[<path>,<attribute_name>]'@<index>']'=<new_value>⁽²⁾</code>		
<code>prompt [<prompt_specifier>]</code>	Exec Config	Customizes the prompt using variables \$default, \$dn, \$mode, \$nodename, \$rdn, and \$user or any desired string or combination of these variables for the lifetime of the CLI session. For details, see Table 7.
<code><RDN></code>	Exec Config	Changes the CLI position to the RDN. If the MO does not exist, the MO is created in Config mode. For details, see Section 3.6 Create MO on page 99.
<code>reinit [<path>]</code>	Config	Resets an MO, an attribute, or a struct member to its initial state. For details, see Section 3.7 Reinitialize MO on page 102.
<code>reinit [<path>,<attribute_name>]</code>		
<code>scriptmode [--on --off]</code>	Exec Config	<p>Without parameter, displays the present state of the scriptmode in the ongoing CLI session.</p> <p>Parameter <code>--on</code> turns on scriptmode, in which help function, auto-completion, case correction, and page break is disabled in ongoing CLI session if not done.</p> <p>Parameter <code>--off</code> turns off scriptmode, in which help function, auto-completion, case correction, and page break is enabled back in ongoing CLI session if not done already.</p>



Table 16 Summary of CLI Commands

<pre>show [-r --recursive] [-s --sort] [-v --verbose] [<path>,<attribute_name>] [<struct_member_name>]</pre>	Exec Config	Displays the system configuration and state information as MO properties with the following options:
<pre>show [-r --recursive] [-s --sort] [-v --verbose] [<path>,<attribute_name>]['@ <index>']' ⁽²⁾</pre>		<ul style="list-style-type: none"> • -c --condition – Filters the existing MO instances and displays information for MO instances fulfilling the specified condition. • -m --moc – Specifies the MOC name whose instances are to be filtered. For details, see Section 3.3.9.1 Parameters to Filter MO Information on page 63. • -p --property – Displays attributes under all the MO instances of the specified MOC. • -r --recursive – Displays child MO instances in a recursive manner. • -s --sort – Displays child MO instances sorted according to their instance names. • -v --verbose – Displays all attributes. • <index> – Specifies a sequence element in a sequence attribute. The index starts from 1. • <path> – Is either an LDN or an RDN. For details, see Section 2.4.2 Local Distinguished Name on page 15 and Section 2.4.3 Relative Distinguished Name on page 16. • Without options, only those attributes are displayed having a value assigned and not a default value.
<pre>show [-s --sort] [<path>] -m --moc <moc_name> [-p --property <attribute_name> [,<attribute_name>] ...] [-r --recursive] [-v --verbose] [-c --condition <condition>]</pre>		<p>For details, see Section 3.3 Display Information on page 48.</p>
<pre>show-config [-s --sort] [-v --verbose] [<path>]</pre>	Exec Config	Displays the output in configuration format in a recursive manner. For details, see Section 3.3.2 Display Configurational Information on page 51.



Table 16 Summary of CLI Commands

<pre>show-counters [<DN>] [-j --pmJob <job_id>] [-v --verbose] [-c --counters <counter>[,<counter>] ...]</pre>	Exec Config	<p>Displays real-time values of PM for one MO instance with the following options:</p> <ul style="list-style-type: none"> • -c --counters – Displays only the selected counters. • -j --pmJobId – Displays only counters associated to one active PM job. The parameter is the ID value of one MO instance of the MOC PmJob. • -v --verbose – Displays verbose information. • <DN> – Selects the MO instance to show counters from. If used, this argument must be given first. If omitted, counters from the current MO are displayed. <p>This is an optional command. It is available only if the ME supports displaying measurements through the CLI. For details, see Section 3.3.9.5 Display PM Measurements on page 71.</p>
--	----------------	---



Table 16 Summary of CLI Commands

show-counters [<DN>] [-j --pmJob <job_id>] [-v --verbose] [-c --counters <counter>[,<counter>] ...] [-m --mo <measObj>] [-g --pmGroup <pmGroupId>[,<pmGroupId>] ...] [-f --format <outputFormat>]	Exec Config	Displays real-time values of PM for one MO instance with the following options: <ul style="list-style-type: none">• -c --counters – Displays only the selected counters.• -j --pmJobId – Displays only counters associated to one active PM job. The parameter is the ID value of one MO instance of the MOC PmJob.• -v --verbose – Displays verbose information.• -m --mo – Displays the counters associated with configured or non-configured measured object provided.• -g --pmGroup – Displays only counters associated with the PM Groups. The parameter is the ID values of PM groups to be displayed.• -f --format – The parameter value defines the format in which the output should be grouped. The parameters allowed are DN-MT, MT-DN, PmGroup-DN-MT, PmGroup-MT-DN. By default DN-MT is used to format the output.• <DN> – Selects the MO instance to show counters from. If used, this argument must be given first. If omitted and if there is no -m option, counters from the current MO are displayed. This is an optional command. It is available only if the ME supports displaying measurements through the CLI. For details, see Section 3.3.9.5 Display PM Measurements on page 71.
show-dn	Exec Config	Displays the user location in the MOM.
show-mib [-v --verbose] [-s --sort] [<path>]	Exec Config	Displays MO instance information. For details, see Section 3.3.8.2 Display MO Instance Information on page 62.



Table 16 Summary of CLI Commands

show-table [-r --recursive] [<path>] -m --moc <moc_name> [-p --property <attribute_name> [: <column_width>][, <attribute_name> [: <column_width>]] ...] [-c --condition <condition>] [-s --sort]	Exec Config	Displays MO information in table format with the following options: <ul style="list-style-type: none"> • -m --moc – Specify the MOC name whose instances to be displayed. For details, see Section 3.3.9.1 Parameters to Filter MO Information on page 63. • -p --property – Displays hidden attributes, include the hidden attributes. • -r --recursive – Implies that the displayed instances can be anywhere below the current position. For details, see Section 3.3.9.3 Display MO Information in Tabular Format on page 67.
top	Exec Config	Changes the CLI position to the root position.
up	Exec Config	Changes the CLI position to the parent MO.
validate	Config	Validates the configuration changes in a transaction. Returns Transaction validation failed! or Transaction is valid!.
version	Exec Config	Displays the CLI version. For details, see Section 3.12 Display CLI Version on page 113.
width [<Width>]	Exec Config	Without parameter, displays the number of CLI output characters printed on a line until the line is broken. Primarily intended for informing the CLI about the actual terminal width when the CLI cannot determine it by itself. Setting the width to a value other than the actual terminal width is not recommended. Parameter Width is a number in the range 0–2147483647. Default is zero, indicating no line break for non-tabular output, and that tabular output uses the actual terminal window size.
(<command> <action>) format	Exec Config	Replaces the string with escape sequences in its input to the actual characters in its output. For more information, see Section 3.13.5 Format Command on page 118.

(1) The syntax '**['<existing_sequence_element>']**' means here that **<existing_sequence_element>** is mandatory in the command.

(2) The syntax '**['@<index>']**' means here that **@<index>** is mandatory in the command.

(3) The syntax '**['@<position>']**' means here that **@<position>** is mandatory in the command.



3.2 Success and Error Indications

The conditions for successful command completion are checked in multiple steps as follows:

- The user must have the proper authorization.
- The command must comply to CLI syntax rules.
- The command must comply to the constraints defined in the model.
- The command must comply to semantic rules, that is, the system state must be valid after the command completion.
- The system must be in a state to be able to process the command.

If the CLI command completes with success, no printout is provided. If the operation fails, an error message is displayed in the following format:

```
<generic_error_message><specific_error_message>
```

The error messages are as follows:

- Generic error message – Error text using the same message format or template for error indication for all the MOCs and attributes of the same properties in the same error situation. Examples of generic error messages are shown in Table 17.
- Specific error messages – Context-specific details are provided by the model.

As an exception, command **validate** returns a printout on success, and actions can return printout if the action return value is not void.

Table 17 Examples of Generic Error Messages

Error Message	Error Semantics
ERROR: Command not found	Invalid command syntax.
ERROR: Can not instantiate system created object	The user tried to create a MOC that is defined as system created.
ERROR: Can not delete system created object	The user tried to delete a MOC that is defined as system created.
ERROR: Attribute '<attribute_name>' is read-only	The user tried to modify an MO attribute that is defined as read only.
ERROR: Attribute '<attribute_name>' is read-only (can't be deleted)	The user tried to delete an MO attribute that is defined as read-only.
ERROR: Parent '<parent_DN>' does not exist	The user tried to create an MO whose parent does not exist.



Table 17 Examples of Generic Error Messages

Error Message	Error Semantics
ERROR: Attribute '<attribute_name>' is restricted	The user tried to modify an MO attribute that is defined as restricted.
ERROR: Call command failed, error code: <error_reason>	Depending on the command and the MO instance, the error reason can be one of the following: <ul style="list-style-type: none"> • ComAborted • ComAlreadyExist • ComCommitFailed • ComFailure • ComInvalidArgument • ComNoResources • ComNotActive • ComNotExist • ComObjectLocked • ComPrepareFailed • ComTimeOut • ComTryAgain • ComValidationFailed • Unknown return code
ERROR: Invalid value <issued command with arguments> for integer parameter <issued command with arguments>. Values are in range [min, max]	Invalid parameter value. The allowed values are in the specified range [min, max].
ERROR: Element not visible	The user cannot access non-visible elements.
ERROR: Instances of <MOC_name> are not creatable	The user tried to create an MO instance that cannot be created.
ERROR: Instances of <MOC_name> are not deletable	The user tried to delete an MO instance that cannot be deleted.
ERROR: Invalid index '0'. The lowest index in a sequence is '1'	The user tried to select a sequence element in a sequence attribute using the invalid index 0. Index values start from 1.
ERROR: Multiple MO classes with same name in different paths: <MO class path> <MO class path> A class path is required	There is more than one MOC with the same name when using parameter -m --moc. The MOC path is a comma-separated list of the class names, for example, ManagedElement=NO DE06ST, SystemFunctions=1, SysM=1, Snmp=1. This MOC path can be used as argument to parameter -m --moc.

3.2.1 Error Codes

The error codes shown in CLI error messages are described in Table 18.



Table 18 Error Codes

Error Code	Description
ComAborted	The function call failed and was ended. The function did not change any persistent data.
ComAlreadyExist	The function call failed, as something that was to be created exists.
ComCommitFailed	The function call failed in the commit phase. Some participants failed to commit and the total transactional result can be inconsistent. A human can be needed to resolve the situation.
ComFailure	The function call failed, as an error occurred that is specific for the function implementation.
ComInvalidArgument	The function call failed, as an argument is invalid.
ComNoResources	The function call failed, as there was no available resource, such as memory.
ComNotActive	The function cannot provide any service, as the service is not started.
ComNotExist	The function call failed, as something sought after did not exist.
ComObjectLocked	The function call failed, as an object is locked.
ComPrepareFailed	The function call failed in the prepare phase of the transaction. The transaction was ended.
ComTryAgain	The function cannot provide any service currently. The problem is temporary and the caller can retry later.
ComValidationFailed	The function call failed, as the data did not validate. This error code is used only as return code from command commit .

3.3 Display Information

This section describes how to display the following information:

- Single-valued attribute, see Section 3.3.1 Display Single-Valued Attribute on page 49
- Configurational information, see Section 3.3.2 Display Configurational Information on page 51
- Struct, see Section 3.3.3 Display Struct on page 53
- Struct member, see Section 3.3.4 Display Struct Member on page 54
- Sequence of simple type, see Section 3.3.5 Display Sequence of Simple Type on page 55
- Sequence of struct, see Section 3.3.6 Display Sequence of Struct on page 56



- Sequence element of sequence of struct, see Section 3.3.7 Display Sequence Element of Sequence of Struct on page 57
- MO instance information, see Section 3.3.8.2 Display MO Instance Information on page 62
- MO information in tabular format, see Section 3.3.9.3 Display MO Information in Tabular Format on page 67
- PM measurements, see Section 3.3.9.5 Display PM Measurements on page 71

3.3.1 Display Single-Valued Attribute

Command `show [-r|--recursive] [-v|--verbose] [<path>,<attribute_name>` displays a single-valued attribute.

A summary on the printout syntax and the displayed information with various attribute types is shown in Table 19.

Table 19 Show Single-Valued Attribute Matrix

Model Property ⁽¹⁾	<code>show [<path>,<attribute_name></code>	<code>show -v --verbose [⁽²⁾<path>,<attribute_name></code>
Read-write attribute with value that is not default value	<code><attribute_name>=<attribute_value></code>	<code><attribute_name>=<attribute_value> ⁽³⁾</code>
Read-only or restricted attribute with value that is not default value	<code><attribute_name>=<attribute_value></code>	<code><attribute_name>=<attribute_value> ⁽⁴⁾</code>
Attribute with value equal to default value	<code><attribute_name>=<attribute_value></code>	<code><attribute_name>=<attribute_value> <default> ⁽⁵⁾</code>
Attributes without value ⁽⁶⁾	<code><attribute_name>=[]</code>	<code><attribute_name>=[]<empty> ⁽³⁾</code>

(1) The description is valid both for attributes and struct members.

(2) The `<key>` printout is present if the struct member is key.

(3) The `<passphrase>` printout is present if the attribute is a passphrase string.

(4) The `<read only>` printout is present if the attribute is read-only.

(5) The `<read only>` printout is also present if the attribute is read-only.

(6) That is, optional or nillable attributes that have no value assigned.

The display formats of supported attribute value data types are shown in Table 20.

Table 20 Display Formats of Supported Attribute Value Data Types

Attribute Value Data Type	Description
bool	Displayed as false or true



Table 20 Display Formats of Supported Attribute Value Data Types

Attribute Value Data Type	Description
enum	The name of the enumeration member is displayed. Example: administrativeState=LOCKED
float	Displayed as decimal numbers. Scientific notation is used when the lexical representation of the value is too long.
int8, int16, int32, int64, uint8, uint16, uint32, uint64	Displayed as integers.
moRef	An internal MO reference is displayed as a string containing the LDN of the referred MO. Example: "ManagedElement=NODE06ST, SystemFunctions=1, MyMo=42". If the referred MO is external, the format depends on the external system.
passphrase string	Displayed as a masked value (*****) or in encrypted form, depending on how the ME is configured. For details, see Section 3.5.5 Change Attribute Defined as Password or Passphrase String on page 95.
password	Displayed as a string in encrypted form. Password change is supported in a special way, see Section 3.5.5 Change Attribute Defined as Password or Passphrase String on page 95.
string	Displayed in double quotation marks. Example: "ABC"

```
>show ManagedElement=NODE06ST,userLabel
```

Example 39 Display Single-Valued Attribute

```
>show ManagedElement=NODE06ST,myEmptyAttribute
```

Example 40 Display Single-Valued Attribute for EcimEmpty

```
>show -r ManagementElement=NODE06ST,MOCex1=1,MOCex5=1,anAttrwWithIntDerivedType
```

Example 41 Display Single-Valued Attribute by Command show -r

```
show ManagementElement=NODE06ST,MOCex2=1,userLabel
```

Example 42 Display Single-Valued Attribute by Using LDN

[illegible]



```
(Snm=1)>show-config
```

Example 45 Display Configuration Output Command for Sequence of Keyless Structs

Command **show-config** with parameter **-v** | **--verbose** displays configuration with explicit position for sequence of keyless structs.

```
(Snm=1)>show-config -v
```

Example 46 Display Configuration Output Command with Explicit Position for Sequence of Keyless Structs

3.3.3 Display Struct

Command **show** **[-v|--verbose]** [**<path>**,] **<attribute_name>** displays attributes defined as struct.

Printout format:

```
<attribute_name>
  <struct_member_name>=<value>
  <struct_member_name>=<value>
  <struct_member_name>=[]
[... ]
  <struct_member_name>=<value>
```

Note: The nillable attributes are displayed as ordinary attributes. See Table 19.



```
(config-M0ex8=1)>show aSimpleStruct
```

Example 47 Display Struct

```
(config-M0ex9=1)>show --verbose aSimpleStruct
```

Example 48 Display Struct with CLI Operation Parameter

```
(config-M0ex10=1)>show -r aSimpleStruct
```

Example 49 Display Struct with CLI Operation Parameter

3.3.4 Display Struct Member

This section provides information about displaying structure member.

3.3.4.1 Display Single-Valued Structure Member

Command `show [-v|--verbose] [<path>,] <attribute_name>, <struct_member_name>` displays single-valued struct members.

The printout is according to the data type of the struct member.

```
(config-M0ex11=1)>show aSimpleStruct, str1
```

Example 50 Display Struct Member

```
(config-M0ex11=1)>show -v aSimpleStruct, int2
```

Example 51 Display Struct Member with CLI Operation Parameter



3.3.4.2 Display Sequence Structure Member

Command **show [-v|--verbose] [<path>,<attribute_name>,<struct_member_name>** displays structure members defined as sequence (also known as multi-valued struct members).

```
(config-structWithMultivalueMembers)>show intMultivalueMember
```

Example 52 Display Sequence Structure Member

```
(config-structWithMultivalueMembers)>show -v intMultivalueMember
```

Example 53 Display Sequence Structure Member with CLI Operation Parameter

3.3.4.3 Display Sequence Element of Sequence Structure Member

Command **show [-v|--verbose] [<path>,<attribute_name>,<struct_member_name>['@<index>']** displays a sequence element of a sequence structure member.

```
(config-structWithMultivalueMembers)>show intMultivalueMember[@2]
```

Example 54 Display Sequence Structure Member Using Positional Index

3.3.5 Display Sequence of Simple Type

Command **show [-r|--recursive] [-v|--verbose] [<path>,<sequence_attribute_name>** displays attributes defined as sequence (also known as multi-valued attributes).

Printout format if the sequence contains elements of the same type:

```
<sequence_attribute_name>
  <sequence_element_value>
  <sequence_element_value>
  [...]
  <sequence_element_value>
```



```
(config-MultivalueThing=1)>show PlainMultivalueAttr  
rs=1,stringMultivalueAttr
```

Example 55 Display Sequence of Strings

```
(config-MultivalueThing=1)>show -r PlainMultivalueAttr  
rs=1,stringMultivalueAttr
```

Example 56 Display Sequence of Strings with CLI Operation Parameter

```
(config-PlainMultivalueAttr=1)>show -v intMultivalueAttr
```

Example 57 Display Sequence of Integers

3.3.5.1

Display Sequence Element of Sequence of Simple Type

Command `show [-v|--verbose] [<path>,<sequence_attribute_name> ['@<index>']` displays a selected sequence element from a sequence of simple type using positional index. The positional index starts from 1.

```
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
```

```
show intMultivalueAttr[@1]
```

Example 58 Display Sequence Element of Sequence of Simple Type

3.3.6

Display Sequence of Struct

Command `show [-r|--recursive] [-v|--verbose] [<path>,<attribute_name>` displays attributes defined as a sequence of struct.

Printout format:



```
<attribute_name>
  <struct_member_name>=<value>
  <struct_member_name>=<value>
  <struct_member_name>=[]
[...]
  <struct_member_name>=<value>
<attribute_name>
  <struct_member_name>=<value>
  <struct_member_name>=<value>
  <struct_member_name>=[]
```

```
>show ManagedElement=NODE06ST,SystemFunctions=1,sysM=1,Snmp=1,agentAddress
```

```

[...]
```

Example 59 Display Sequence of Structs without Key

```
>show -v ManagedElement=NODE06ST,SystemFunctions=1,sysM=1,Snmp=1,agentAddress
```

```

[...]
```

Example 60 Display Sequence of Structs without Key with CLI Operation Parameter

3.3.7

Display Sequence Element of Sequence of Struct

Command `show [-r|--recursive] [-v|--verbose] [<path>,<attribute_name>,<key_struct_member_name>=<key_struct_member_value>` displays selected sequence elements if they contain a struct with a key member.

Printout format:

```
<key_struct_member_name>=<key_struct_member_value>
  <struct_member_name>=<struct_member_value>
  <struct_member_name>=<struct_member_value>
[...]
  <struct_member_name>=<struct_member_value>
```

In Example 61, attribute `multiValueStructWithIntId` is defined as a sequence of struct and the struct has the following members:

- `id` – Defined as `int64`, that is, the key member of the struct
- `name` – Defined as a string with default value `"countryName"`
- `capital` – Defined as a string
- `population` – Defined as `int64` with default value `0`

```
(config-PlainMultivalueAttrs=1)>show -v multiValueStructWithIntId
```

```
show -v PlainMultivalueAttrs=1,⇒
multiValueStructWithIntId=1
```

Example 61 Display Sequence Element of Sequence of Structs with Key

```
>show ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Snmp=1,agentAddress,host
```

Example 62 Display Sequence Element of Sequence of Structs without Key

```
(config-PlainMultivalueAttrs=1)>show -r multiValueStructWithIntId
```

Example 63 Display Sequence Element of Sequence of Structs with Key with CLI Operation Parameter



3.3.8 Display MO Information

This section describe how to display MO information.

3.3.8.1 Display MO Instances

This section describes how to display MO instances.

3.3.8.1.1 MO Instance Sorting

By default, MO instances are displayed in an ME-dependent order, for example, in creation order.

The MO instances can also be displayed in a sorted order, by adding parameter **-s | --sort**. The instances are sorted in the following order:

1. All instances with numeric names are displayed in increasing numeric order.
2. All other instances are displayed in alphabetical order.

Note: Sorting MO instances increases the command execution time. Only MO instances are sorted, not elements of sequence attributes.

3.3.8.1.2 Display Single MO

Command **show [-s|--sort] [-v|--verbose] [<path>]** displays a single MO.

Printout format:

```
<moc_name>=<key_value>
  <attribute>
  <attribute>
[...]
```

```
<struct>
<struct>
[...]
```

```
<child_moc_name>=<key_value>
<child_moc_name>=<key_value>
[...]
```

The list of actions for a class is deleted from the show output.

```
>show ManagedElement=N0DE06ST, SystemFunctions=1, SysM=1, Sntp=1
```

```


```

Example 64 Display Single MO



3.3.8.1.3 Display Single MO and Its Child MOs

Command **show [-s|--sort] [-r|--recursive] [-v|--verbose] [<path>]** displays a single MO and its child MOs.

The displayed information is according to the single MO printout for the root MO and also for the child MOs. That is, all child MOs with all their properties, excluding the actions, are displayed in a recursive way.

```
>show -r ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Snmp=1
```

Example 65 Display Single MO and Its Child MOs

3.3.8.1.4 Display Single MO and Its Child MOs in Configuration Printout Format

Command **show-config [-s|--sort] [-v|--verbose] [<path>]** displays a single MO and its child MOs in configuration format.

The displayed information is according to the recursive MO printout, with the addition of navigation command **up** in specific positions. The displayed information forms a valid CLI command sequence that can be input for the CLI. The information allows configuration data export and import by copy/paste.

```
>show-config ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Snmp=1
```

Example 66 Display Single MO and Its Child MOs in Configuration Printout Format

3.3.8.1.5 Summary of Displayed MO Properties

A summary of the elements displayed by different display options shown in Table 21.



Table 21 Show MO Instance Matrix

Model Property	show <path>	show -r --recursive <path>	show-config <path>	show -v --verbose <path>	show-config -v --verbose <path>
System-created child MOs	Yes	Yes	Yes ⁽¹⁾	Yes	Yes ⁽¹⁾
Not creatable child MOs	Yes	Yes	Yes ⁽¹⁾	Yes	Yes ⁽¹⁾
Child MOs recursively	No	Yes	Yes	No	Yes
Read-write attribute with value that is not default value	Yes	Yes	Yes	Yes	Yes
Read-only or restricted attribute with value that is not default value	Yes	Yes	No	Yes	No
Restricted attribute with value that is not default value	No ⁽²⁾	No	No	Yes	No
Attribute with value equal to default value	No	No	No	Yes	Yes
Attributes without value ⁽³⁾	No	No	No	Yes	Yes
Action names	No	No	No	No	No

(1) Only if the MO subtree has configurable elements.

(2) This is present in the show attribute printout.

(3) That is, optional or nillable attributes with no values assigned.

The command is atomic and does not include non-committed changes in other transactions.

```
>show -v ManagedElement=NODE06ST,SystemFunctions=1,Fm=1
```

1. Was ist
 2. Kanten-Mengenformale Notation
 3. Kanten-Mengenformale Notation
 4. Kanten-Mengenformale Notation
 5. Kanten-Mengenformale Notation
 6. Kanten-Mengenformale Notation
 7. Kanten-Mengenformale Notation
 8. Kanten-Mengenformale Notation
 9. Kanten-Mengenformale Notation
 10. Kanten-Mengenformale Notation

Example 67 Display MO

3.3.8.2 Display MO Instance Information

Command **show-mib [-v|--verbose] [-s|--sort] [<path>]** displays the list of instance information without their contents. The MO instance names are displayed recursively.

With parameter `s | --sort`, MO instances are sorted according to their instance names.

```
(config-SecM=1)>show-mib
```

Page 10 of 10

Example 68 Display MO Instance Information Recursively

Command `show-mib` with parameter `-v | --verbose` displays the complete path of the DN.

```
(config-ManagedElement=N0DE06ST)>show-mib -v M0Cex5=1
```

[illegible]

Example 69 Display MO Instance Information with CLI Operation Parameter



3.3.9 Filter MO Information

The MO information can be reduced with the filter options added to commands **show** and **show-table**. The filtered information is displayed in tree structure and in tabular format.

3.3.9.1 Parameters to Filter MO Information

The mandatory parameter **-m** | **--moc** signifies the name of a child class (MOC) under the current context where the user is located.

If there is more than one class with same name, the name must be qualified by prepending the class name with the parent class name and a comma. For example, assume that there is a class *Xyz* under *SystemFunctions* and another class with the same name under *ManagedElement*. These classes can then be addressed by *SystemFunctions,Xyz* and *ManagedElement,Xyz*, respectively.

Optional parameters:

- **-p** | **--property** signifies the attributes that the user wants to display under the MOC, specified in option **-m** | **--moc**. The attribute list is specified by [**-p** **<attribute_name>**], where attributes are separated by a comma. The list can include zero to many attributes, where zero means that no attribute is printed.
- For command **show-table**, which shows MO information in tabular format, the column width can optionally be specified for attributes by using the syntax **-p** **<attribute_name:column_width>**.
- To filter and display MO information based on a condition, use option **-c** | **--condition**.

If the attribute used in the condition is of the type string, the value specified for that attribute must be quoted, else error message *Condition String not quoted* is displayed.

Syntax for condition:

```
[Condition]=[<attribute><operator><value>]
```

Valid operators:

- **==** or **=** for left-hand side is equal to the right-hand side
- **>=** for left-hand side is greater than or equal to the right-hand side
- **<=** for left-hand side is less than or equal to the right-hand side
- **>** for left-hand side is greater than the right-hand side
- **<** for left-hand side is smaller than the right-hand side
- **<>** for left-hand side is greater or less than (not equal) the right-hand side



- `=~` for left-hand side matching the regular expression on the right-hand side.

To specify multiple conditions, separate them by the logical operator **OR (||)** or **AND (&&)**. To negate the expression, let it be preceded by a **NOT**. To group expressions, use parentheses.

Multiple condition syntax:

```
[Condition]=((condition)&&(condition))⇒  
=((condition)|| (condition))=NOT((condition))
```

Values are entered in the same format as when setting values. Strings must be quoted if they contain spaces or special characters. A special value is `[]`, which is the empty value.

When matching a regular expression, the left-hand side attribute value is converted to a string, as if the value would be printed. This includes structs, with the special detail that struct members are converted to a sequence of space-separated `<member>=<value>` statements. Password types cannot be matched with a regular expression.

When the left-hand side attribute is a sequence attribute, a condition matches if at least one value in the sequence matches.

3.3.9.2 Search for Information in MO Tree Structure

Command `show [-s|--sort] [<path>] -m|--moc <moc_name> [-p|--property <attribute_name> [,<attribute_name>]...] [-r|--recursive] [-v|--verbose] [-c|--condition <condition>]` searches for information in an MO tree structure.

For information about the filtering parameters `-m|--moc`, `-p|--property`, and `-c|--condition`, see Table 16.

Parameter `-r|--recursive` can be used to search recursively in the whole MO tree. When the parameter is used, the DN for each matching MO instance is displayed. Omitting the parameter implies searching of one level and the RDN for each matching MO instance is displayed.

With parameter `-s|--sort`, MO instances are sorted according to their instance names.



```
>show SysM=1 -m Schema -r
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ECIM_CommonLibrary
  baseModelIdentifier="ECIM_CommonLibrary"
  baseModelVersion="1.2"
  identifier="ECIM_CommonLibrary"
  version="1.2"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComTop
  baseModelIdentifier="ECIM_Top"
  baseModelVersion="2.1.0"
  identifier="ComTop"
  version="10.10.0"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSecM
  baseModelIdentifier="ECIM_Security_Management"
  baseModelVersion="2.0"
  identifier="ComSecM"
  version="11.0.1"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLocalAuthorization
  baseModelIdentifier="ECIM_Local_Authorization"
  baseModelVersion="2.0.0"
  identifier="ComLocalAuthorization"
  version="0.11.1"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLdapAuthentication
  baseModelIdentifier="ECIM_LDAP_Authentication"
  baseModelVersion="2.0"
  identifier="ComLdapAuthentication"
  version="11.0.0"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSysM
  baseModelIdentifier="ECIM_SysM"
  baseModelVersion="3.1.0"
  identifier="ComSysM"
  version="3.1.0"
```

Example 70 Display Specific MO Type in Tree Structure

Display Specific Attributes, or No Attributes, for Found MO Instances

Command `show` with parameter `-p|--property` displays specified attributes under all the MO instances of the specified MOC.

```
>show SysM=1 -m Schema -r -p
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ECIM_CommonLibrary
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComTop
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSecM
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLocalAuthorization
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLdapAuthentication
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSysM
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComFm
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSnmp
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComFileM
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=CmwPm
```

Example 71 Display only Found Instances, no Attributes



```
(config-SystemFunctions=1)>show -r SysM=1 -m Schema -p
identifier,baseModelVersion
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ECIM_CommonLibrary
  identifier="ECIM_CommonLibrary"
  baseModelVersion="1.2"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComTop
  identifier="ComTop"
  baseModelVersion="2.1.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComSecM
  identifier="ComSecM"
  baseModelVersion="2.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComLocalAuthorization
  identifier="ComLocalAuthorization"
  baseModelVersion="2.0.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComLdapAuthentication
  identifier="ComLdapAuthentication"
  baseModelVersion="2.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComSysM
  identifier="ComSysM"
  baseModelVersion="3.1.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComFm
  identifier="ComFm"
  baseModelVersion="4.0.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComSnmp
  identifier="ComSnmp"
  baseModelVersion="1.2"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComFileM
  identifier="ComFileM"
  baseModelVersion="3.1.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=CmwPm
  identifier="CmwPm"
  baseModelVersion="1.2"
```

Example 72 Displaying Specific Attributes under All Existing MO Instances for a MOC

Display Attributes Based on a Condition

Command **show** with option **-c | --condition** filters the existing MO instances and displays information for MO instances fulfilling the specified condition.

```
>show -r -m Schema -c version==11.0.0
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComLdapAuthentication
  baseModelIdentifier="ECIM_LDAP_Authentication"
  baseModelVersion="2.0"
  identifier="ComLdapAuthentication"
  version="11.0.0"
ManagedElement=NODE06ST,SystemFunctions=1,SysM=1,Schema=ComFileM
  baseModelIdentifier="ECIM_FileM"
  baseModelVersion="3.1.0"
  identifier="ComFileM"
  version="11.0.0"
```

Example 73 Specify Condition to Filter MO Information

```
(config-SystemFunctions=1)>show SysM=1 -m Schema -p identifier,baseModelIdentifier -c version==11.0.0
Schema=ComLdapAuthentication
  identifier="ComLdapAuthentication"
  baseModelIdentifier="ECIM_LDAP_Authentication"
Schema=ComFileM
  identifier="ComFileM"
  baseModelIdentifier="ECIM_FileM"
```

Example 74 Specify Attributes and Condition for Filtering MO Information



```
(config-SysM=1)>show -m Schema -p identifier,baseModelIdentifier
-c version==3.1.0 && baseModelVersion==3.1.0
Schema=ComSysM
  identifier="ComSysM"
  baseModelIdentifier="ECIM_SysM"
```

Example 75 Specify Attributes and Multiple Conditions for Filtering MO Information

Regular Expression Search

For examples of regular expression searching, see Example 76 through Example 78.

```
(config-SysM=1)>show -r -m SnmpTargetV3 -c operationalState=~EN
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1, SnmpTargetV3=1
  address="127.0.0.1"
  informRetryCount=3
  informTimeout=10
  operationalState=ENABLED
```

Example 76 Regular Expression Search

```
>show -r -m Snmp -c agentAddress=~"port=161"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1
  administrativeState=LOCKED
  engineId="1234567890"
  operationalState=DISABLED
  agentAddress
    host="0.0.0.0"
    port=161
```

Example 77 Regular Expression Search with a struct

```
>show -r -m Snmp -c agentAddress=~"host=0\.\0\.\0 port=161"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1
  administrativeState=LOCKED
  engineId="1234567890"
  operationalState=DISABLED
  agentAddress
    host="0.0.0.0"
    port=161
```

Example 78 Regular Expression Search with Multiple struct Members

3.3.9.3

Display MO Information in Tabular Format

Command `show-table [-r|--recursive] [<path>] -m|--moc <moc_name> [-p|--property <attribute_name> [:<column_width>][,<attribute_name>[:<column_width>]] ...] [-c|--condition <condition>] [-s|--sort]` displays the MO information based on a MOC in a tabular format. It displays the show verbose printout without metadata, for example, no tags are displayed in the table. The command does not display struct attributes and multivalued attributes.

The user can choose information based on attribute selection and can also specify the column width.

If the MOC contains hidden attributes, the command does not display them. To display hidden attributes, including the hidden attributes, use option `-p|--property`



With parameter **s** | **--sort**, MO instances are sorted according to their instance names.

```
(Pm=1)>show-table -m PmJob -p currentJobState,jobPriority,jobType,pmJobId
```

currentJobState	jobPriority	jobType	pmJobId
ACTIVE	MEDIUM	MEASUREMENTJOB	POT_15min_Job
ACTIVE	LOW	MEASUREMENTJOB	PU_15min_Job
STOPPED	MEDIUM	THRESHOLDJOB	VM_15min_Job

Example 79 Display MO Information in Table

When using parameter **-r** | **--recursive**, a recursive search is done for the MOC. The parent MO LDN is displayed above the table. If there are instances under different parents, a new table is displayed for each parent.

Display Attributes with Show-Table

Command **show-table** with option parameter **-p** | **--property** displays the specific attributes under the MOC.

The width of the column can be specified, as shown in Example 80 and Example 81.

```
(config-ManagedElement=NODE06ST)>show-table -m
AMoc -p userLabel:20
=====
|userLabel      |
=====
|UserLabelValue |
=====
```

Example 80 Display Single Attribute with Command show-table

```
(config-ManagedElement=NODE06ST)>show-table -m AMoc
-p MOCexId, userLabel
=====
|MOCexId | userLabel      |
=====
|1       | UserLabelValue |
=====
```

Example 81 Display Multiple Attributes with Command show-table

If no value is available for an attribute, it displays as empty, as shown in Example 82.

```
(config-SystemFunctions=1)>show-table -m FileM
=====
|fileMId | userLabel |
=====
|1       |          |
=====
```

Example 82 Display Empty Attribute with Command show-table

Display Attributes in Tabular Format Based on a Condition

Example 83 shows how to display attributes based on a condition.



```
(config-MOCex6=1)>show-table -m M0ex9 -p administrativeState,defaultValue -c (defaultValue==0)
=====
| administrativeState | defaultValue |
| UNLOCKED           | 0           |
=====
```

Example 83 Display Attributes Based on Condition with Command show-table

Show-Table Width and Column Compression

The width of each column is automatically adjusted to the width of the content. Both the column heading, that is, the attribute name, and the content of each row are considered. If a table contains more than 100 rows, only the content of the first 100 rows are used to compute the widths.

If the terminal width is not enough to display all columns with their optimal widths, the columns are compressed. When the content of a table cell is too wide to be displayed in its entirety, it is compressed by replacing the middle part of the content with three dots, "...". For instance, MEASUREMENTJOB is compressed to MEAS . . . JOB, if the column is compressed to 10 characters.

If the column width for an attribute has been specified, it is used as the minimum width for that attribute. That is, if the table needs to be compressed, the column is not compressed to fewer characters than the specified width.

If the table is not possible to display within the current terminal width based on specified column widths, show-table returns an error message.

3.3.9.4

Auto-Completion of Attributes under Filter Options

Command **show** and **show-table** with option parameter **-p|--property** auto-completes the list of attributes under a MOC, specified with parameter **-m|--moc**. The hidden attributes cannot be displayed in the auto-completion list.

```
(config-SysM=1)>show -m Snmp -p_
```

Example 84 Completion Possibility List for Attributes for Command show

For command **show-table**, the multi-value and struct attributes are not be listed in the auto-completion list under option **-p | --property**.

```
(config-SysM=1)>show-table -m Snmp -p
```

Example 85 Completion Possibility List for Attribute for Command show-table

Note: Auto-completion for attributes under option parameter `-c|--condition` is not supported for commands `show` and `show-table`.

3.3.9.5 Display PM Measurements

Command **show-counters** is an optional command. It is available only if the ME supports displaying measurements through the CLI.

The support from the ME can also be at two different levels, resulting in that the two different versions of the command are possible. Version one supports displaying measurements for Managed Objects, and allows selection of measurements according to the PM jobs they belong to. Version two supports displaying measurements also for other Measured Objects that are not Managed Objects, and also allows selection of measurements according to the PM Groups they belong to.

Version two also allows selection of how the output needs to be grouped, and allows wildcards in selection criteria. When using wildcards, the command response time can be significantly long, so filter criteria to the command needs to be wisely chosen to avoid long waiting time.

Note: The command can be cancelled at any time by pressing **Ctrl+C**.

The `static help` command output provides information on which version of `show-counters` commands that is active, if any.

The syntax for version 1 of `show-counters` is:

```
show-counters [<DN>] [-j|--pmJob <job_id>] [-v|--verbose]
[-c|--counters <counter>[,<counter>] ...]
```



The syntax for version 2 of **show-counters** is:

```
show-counters [<DN>] [-j|--pmJob <job_id>] [-v|--verbose]  
[-c|--counters <counter>[,<counter>] ...] [-m|--mo <measObj>]  
[-g|--pmGroup <pmGroupId>[,<pmGroupId>] ...] [-f|--format  
<printoutFormat>]
```

The command **show-counters** displays active PM counters for a Measured Object. The command works analogous to the other **show** commands, but displays PM counters associated with Measured Objects instead of attributes. Also, unlike multi-valued attributes, multi-valued counters are displayed one after each other on the same line, with comma as delimiter.

```
(MeasObj=1)>show-counters
```

Example 86 Display PM Measurements with Command **show-counters**

```
(MeasObj=1)>show-counters ManagedElement=1,MeasObj=1
```

Example 87 Display PM Measurements with Command **show-counters <DN>**

```
(MeasObj=1)>show-counters -v
```

Example 88 Display PM Measurements with Command **show-counters -v**

The number of displayed counters can be decreased by filtering out specific counters by name or from a selected PM job.

If there is a delay in fetching the measurements when version2 of **show-counters** command is executed, dots are printed on the CLI terminal until a response is received as shown in Example 89 and the command can be cancelled at any time by pressing **Ctrl+C**.



```
>show-counters -m M01TryAgain3
```

Example 89 Display Dots while Executing show-counters

3.3.9.5.1 Filtering options for show-counters

The number of displayed counters can be decreased by filtering out specific counters by name, PM job ids, PM group ids, measured object, or a combination of one or more of the mentioned. If no measured object/DN is mentioned explicitly, then the cursor location is considered as measured object. If both <DN> and -m options are specified, then preference is given to -m.

```
(MeasObj=1)>show-counters -c floatCounterNonActive floatCounterNonActive=[]
(MeasObj=1)>show-counters -v -c floatCounterActive,
                        someOtherCounter --pmJob=aJob
```

Example 90 Display Specific Counters

```
>show-counters -m M01
```

Example 91 Display Counters Specific to a Measured Object

```
>show-counters -g group0,group1
```

Example 92 Display Counters Specific to a PM Groups

Measurements of one or more group-Ids can be fetched by providing * in input as shown in the Example 93. All group names which starts with group are displayed in the output.



```
>show-counters -g group* -f PMGroup-DN-MT
```

Example 93 Display Counters Using Filter Construct '*' with group id in input.

Measurements of one or more measured objects can be fetched by providing * in input as shown in the Example 94. All measured objects which starts with MoAdd are displayed in the output.

```
>show-counters -m MoAdd*
```

Example 94 Display Counters Using Filter Construct '*' with Measured Object in Input

Measurements of one or more measurement types can be fetched by providing * in input as shown in the Example 95. All measurement types which starts with aSuspectPDF are displayed in the output.

```
>show-counters -c aSuspectPDF*
```

Example 95 Display Counters Using '*' with Measurement Type in Input

3.3.9.5.2 Formatting the Output

The output can be formatted by using the format option -f / --format.

- DN-MT – Group by measured object then measurement type.
- MT-DN – Group by measurement type then measured object.
- PMGroup-DN-MT – Group by PM group then measured object then measurement type.
- PMGroup-MT-DN – Group by PM group then measurement type then measured object.

2004
 2005
 2006
 2007
 2008
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 2024
 2025
 2026
 2027
 2028
 2029
 2030
 2031
 2032
 2033
 2034
 2035
 2036
 2037
 2038
 2039
 2040
 2041
 2042
 2043
 2044
 2045
 2046
 2047
 2048
 2049
 2050
 2051
 2052
 2053
 2054
 2055
 2056
 2057
 2058
 2059
 2060
 2061
 2062
 2063
 2064
 2065
 2066
 2067
 2068
 2069
 2070
 2071
 2072
 2073
 2074
 2075
 2076
 2077
 2078
 2079
 2080
 2081
 2082
 2083
 2084
 2085
 2086
 2087
 2088
 2089
 2090
 2091
 2092
 2093
 2094
 2095
 2096
 2097
 2098
 2099
 2100
 2101
 2102
 2103
 2104
 2105
 2106
 2107
 2108
 2109
 2110
 2111
 2112
 2113
 2114
 2115
 2116
 2117
 2118
 2119
 2120
 2121
 2122
 2123
 2124
 2125
 2126
 2127
 2128
 2129
 2130
 2131
 2132
 2133
 2134
 2135
 2136
 2137
 2138
 2139
 2140
 2141
 2142
 2143
 2144
 2145
 2146
 2147
 2148
 2149
 2150
 2151
 2152
 2153
 2154
 2155
 2156
 2157
 2158
 2159
 2160
 2161
 2162
 2163
 2164
 2165
 2166
 2167
 2168
 2169
 2170
 2171
 2172
 2173
 2174
 2175
 2176
 2177
 2178
 2179
 2180
 2181
 2182
 2183
 2184
 2185
 2186
 2187
 2188
 2189
 2190
 2191
 2192
 2193
 2194
 2195
 2196
 2197
 2198
 2199
 2200
 2201
 2202
 2203
 2204
 2205
 2206
 2207
 2208
 2209
 2210
 2211
 2212
 2213
 2214
 2215
 2216
 2217
 2218
 2219
 2220
 2221
 2222
 2223
 2224
 2225
 2226
 2227
 2228
 2229
 2230
 2231
 2232
 2233
 2234
 2235
 2236
 2237
 2238
 2239
 2240
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254
 2255
 2256
 2257
 2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286
 2287
 2288
 2289
 2290
 2291
 2292
 2293
 2294
 2295
 2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304
 2305
 2306
 2307
 2308
 2309
 2310
 2311
 2312
 2313
 2314
 2315
 2316
 2317
 2318
 2319
 2320
 2321
 2322
 2323
 2324
 2325
 2326
 2327
 2328
 2329
 2330
 2331
 2332
 2333
 2334
 2335
 2336
 2337
 2338
 2339
 2340
 2341
 2342
 2343
 2344
 2345
 2346
 2347
 2348
 2349
 2350
 2351
 2352
 2353
 2354
 2355
 2356
 2357
 2358
 2359
 2360
 2361
 2362
 2363
 2364
 2365
 2366
 2367
 2368
 2369
 2370
 2371
 2372
 2373
 2374
 2375
 2376
 2377
 2378
 2379
 2380
 2381
 2382
 2383
 2384
 2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397
 2398
 2399
 2400
 2401
 2402
 2403
 2404
 2405
 2406
 2407
 2408
 2409
 2410
 2411
 2412
 2413
 2414
 2415
 2416
 2417
 2418
 2419
 2420
 2421
 2422
 2423
 2424
 2425
 2426
 2427
 2428
 2429
 2430
 2431
 2432
 2433
 2434
 2435
 2436
 2437
 2438
 2439
 2440
 2441
 2442
 2443
 2444
 2445
 2446
 2447
 2448
 2449
 2450
 2451
 2452
 2453
 2454
 2455
 2456
 2457
 2458

[illegible]

1. **Abstract**
 2. **Introduction**
 3. **Methods**
 4. **Results**
 5. **Discussion**
 6. **Conclusion**
 7. **References**
 8. **Appendix**
 9. **Tables**
 10. **Figures**
 11. **Supplementary Materials**
 12. **Notes**
 13. **References**
 14. **Appendix**
 15. **Tables**
 16. **Figures**
 17. **Supplementary Materials**
 18. **Notes**
 19. **References**
 20. **Appendix**
 21. **Tables**
 22. **Figures**
 23. **Supplementary Materials**
 24. **Notes**
 25. **References**
 26. **Appendix**
 27. **Tables**
 28. **Figures**
 29. **Supplementary Materials**
 30. **Notes**
 31. **References**
 32. **Appendix**
 33. **Tables**
 34. **Figures**
 35. **Supplementary Materials**
 36. **Notes**
 37. **References**
 38. **Appendix**
 39. **Tables**
 40. **Figures**
 41. **Supplementary Materials**
 42. **Notes**
 43. **References**
 44. **Appendix**
 45. **Tables**
 46. **Figures**
 47. **Supplementary Materials**
 48. **Notes**
 49. **References**
 50. **Appendix**
 51. **Tables**
 52. **Figures**
 53. **Supplementary Materials**
 54. **Notes**
 55. **References**
 56. **Appendix**
 57. **Tables**
 58. **Figures**
 59. **Supplementary Materials**
 60. **Notes**
 61. **References**
 62. **Appendix**
 63. **Tables**
 64. **Figures**
 65. **Supplementary Materials**
 66. **Notes**
 67. **References**
 68. **Appendix**
 69. **Tables**
 70. **Figures**
 71. **Supplementary Materials**
 72. **Notes**
 73. **References**
 74. **Appendix**
 75. **Tables**
 76. **Figures**
 77. **Supplementary Materials**
 78. **Notes**
 79. **References**
 80. **Appendix**
 81. **Tables**
 82. **Figures**
 83. **Supplementary Materials**
 84. **Notes**
 85. **References**
 86. **Appendix**
 87. **Tables**
 88. **Figures**
 89. **Supplementary Materials**
 90. **Notes**
 91. **References**
 92. **Appendix**
 93. **Tables**
 94. **Figures**
 95. **Supplementary Materials**
 96. **Notes**
 97. **References**
 98. **Appendix**
 99. **Tables**
 100. **Figures**
 101. **Supplementary Materials**
 102. **Notes**
 103. **References**
 104. **Appendix**
 105. **Tables**
 106. **Figures**
 107. **Supplementary Materials**
 108. **Notes**
 109. **References**
 110. **Appendix**
 111. **Tables**
 112. **Figures**
 113. **Supplementary Materials**
 114. **Notes**
 115. **References**
 116. **Appendix**
 117. **Tables**
 118. **Figures**
 119. **Supplementary Materials**
 120. **Notes**
 121. **References**
 122. **Appendix**
 123. **Tables**
 124. **Figures**
 125. **Supplementary Materials**
 126. **Notes**
 127. **References**
 128. **Appendix**
 129. **Tables**
 130. **Figures**
 131. **Supplementary Materials**
 132. **Notes**
 133. **References**
 134. **Appendix**
 135. **Tables**
 136. **Figures**
 137. **Supplementary Materials**
 138. **Notes**
 139. **References**
 140. **Appendix**
 141. **Tables**
 142. **Figures**
 143. **Supplementary Materials**
 144. **Notes**
 145. **References**
 146. **Appendix**
 147. **Tables**
 148. **Figures**
 149. **Supplementary Materials**
 150. **Notes**
 151. **References**
 152. **Appendix**
 153. **Tables**
 154. **Figures**
 155. **Supplementary Materials**
 156. **Notes**
 157. **References**
 158. **Appendix**
 159. **Tables**
 160. **Figures**
 161. **Supplementary Materials**
 162. **Notes**
 163. **References**
 164. **Appendix**
 165. **Tables**
 166. **Figures**
 167. **Supplementary Materials**
 168. **Notes**
 169. **References**
 170. **Appendix**
 171. **Tables**
 172. **Figures**
 173. **Supplementary Materials**
 174. **Notes**
 175. **References**
 176. **Appendix**
 177. **Tables**
 178. **Figures**
 179. **Supplementary Materials**
 180. **Notes**
 181. **References**
 182. **Appendix**
 183. **Tables**
 184. **Figures**
 185. **Supplementary Materials**
 186. **Notes**
 187. **References**
 188. **Appendix**
 189. **Tables**
 190. **Figures**
 191. **Supplementary Materials**
 192. **Notes**
 193. **References**
 194. **Appendix**
 195. **Tables**
 196. **Figures**
 197. **Supplementary Materials**
 198. **Notes**
 199. **References**
 200. **Appendix**
 201. **Tables**
 202. **Figures**
 203. **Supplementary Materials**
 204. **Notes**
 205. **References**
 206. **Appendix**
 207. **Tables**
 208. **Figures**
 209. **Supplementary Materials**
 210. **Notes**
 211. **References**
 212. **Appendix**
 213. **Tables**
 214. **Figures**
 215. **Supplementary Materials**
 216. **Notes**
 217. **References**
 218. **Appendix**
 219. **Tables**
 220. **Figures**
 221. **Supplementary Materials**
 222. **Notes**
 223. **References**
 224. **Appendix**
 225. **Tables**
 226. **Figures**
 227. **Supplementary Materials**
 228. **Notes**
 229. **References**
 230. **Appendix**
 231. **Tables**
 232. **Figures**
 233. **Supplementary Materials**
 234. **Notes**
 235. **References**
 236. **Appendix**
 237. **Tables**
 238. **Figures**
 239. **Supplementary Materials**
 240. **Notes**
 241. **References**
 242. **Appendix**
 243. **Tables**
 244. **Figures**
 245. **Supplementary Materials**
 246. **Notes**
 247. **References**
 248. **Appendix**
 249. **Tables**
 250. **Figures**
 251. **Supplementary Materials**
 252. **Notes**
 253

75

```
>show-counters -f PMGroup-MT-DN
```

1. $\text{H}^1(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^n$
 2. $\text{H}^2(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)}{2}}$
 3. $\text{H}^3(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)}{6}}$
 4. $\text{H}^4(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)}{24}}$
 5. $\text{H}^5(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)}{120}}$
 6. $\text{H}^6(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)(n-5)}{720}}$
 7. $\text{H}^7(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)(n-5)(n-6)}{5040}}$
 8. $\text{H}^8(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)(n-5)(n-6)(n-7)}{40320}}$
 9. $\text{H}^9(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)(n-5)(n-6)(n-7)(n-8)}{362880}}$
 10. $\text{H}^{10}(\mathbb{R}^n, \mathbb{R}) \cong \mathbb{R}^{\frac{n(n-1)(n-2)(n-3)(n-4)(n-5)(n-6)(n-7)(n-8)(n-9)}{3628800}}$

Example 99 Display Measurements Using PMGroup-MT-DN Filtering

Table 22 lists the error messages for **show-counters** command

Table 22 Error Messages for show-counters

Error Message	Description
ERROR: Not authorized to execute show-counters	Error is displayed when user is not authorized to execute show-counters command.
ERROR: Measured object <measuredObject> does not exist	Error is displayed when the given measured object does not exist.
ERROR: Invalid argument	Error is displayed when the input PM group ID (or) PM job ID (or) counter/MeasurementType name is invalid.
ERROR: No counter found	Error is displayed when there are no counters available.

3.3.10 Error Indication when Displaying MO Information

If a command that displays information from multiple MO instances fails to retrieve information from a certain MO instance, or group of MO instances, an error message is displayed. Then, the command continues to display information from the following MO instances, if any. After the last MO instance has been displayed, an final error message is also displayed.

The error messages are always displayed left aligned, independent of how deep in the MO hierarchy the affected MO instance is. Example 100 shows how the error messages from `show`, `show-config` and `show-mib` can be displayed.



```
(config-SysM=1)>show -m Schema
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ECIM_CommonLibrary
  baseModelIdentifier="ECIM_CommonLibrary"
  baseModelVersion="1.2"
  identifier="ECIM_CommonLibrary"
  version="1.2"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComTop
ERROR: Failed to get attribute value(s) for ManagedElement=NODE06ST, SystemFunctions=1, SysM=1,
  Schema=ComTop
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComSecM
  baseModelIdentifier="ECIM_Security_Management"
  baseModelVersion="2.0"
  identifier="ComSecM"
  version="11.0.1"
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLocalAuthorization
ERROR: Failed to get attribute value(s) for ManagedElement=NODE06ST, SystemFunctions=1, SysM=1,
  Schema=ComLocalAuthorization
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Schema=ComLdapAuthentication
  baseModelIdentifier="ECIM_LDAP_Authentication"
  baseModelVersion="2.0"
  identifier="ComLdapAuthentication"
  version="11.0.0"
ERROR: Failed to read instances of class Schema under ManagedElement=1, SystemFunctions=1, SysM=1
ERROR: Failed to retrieve all requested information
```

Example 100 Display MO Instance Information Recursively with Error Messages

The show-table command is an exception. It always displays any error messages after the table. See Example 101.

```
(Pm=1)>show-table -m PmJob -p currentJobState, jobPriority, jobType, pmJobId
=====
| currentJobState | jobPriority | jobType          | pmJobId          |
=====
| ACTIVE          | MEDIUM     | MEASUREMENTJOB  | POT_15min_Job   |
=====
ERROR: Failed to get attribute value(s) for ManagedElement=1, SystemFunctions=1, Pm=1,
  PmJob=PU_15min_Job
ERROR: Failed to read instances of class PmJob under ManagedElement=1, SystemFunctions=1, Pm=1
ERROR: Failed to retrieve all requested information
```

Example 101 Display MO Information in Table with Errors

3.4 Change and Display the Position in MO Tree

The MO position can be changed by navigation commands in both Exec mode and Config mode in the following ways:

- By entering the <path>. For example, to change position from root to ManagedElement and then to SysM:

```
(config)>ManagedElement=NODE06ST
(config-ManagedElement=NODE06ST)>SystemFunctions=1, =>
SysM=1
(config-SysM=1)>
```

By entering RDNs, the CLI position can be changed only to child MO instances, not to parent MO instances. The reason is that RDNs are interpreted relative to the CLI position.

- By entering the LDN. For example, to change position from SysM to SystemFunctions:



```
(config-SysM=1)>ManagedElement=NODE06ST, SystemFunctions=1  
(config-SystemFunctions=1)>
```

- By entering command **dn** followed by an LDN, to navigate to an MO instance. The difference between this command and navigation by just entering an LDN is that command **dn** does not create an MO instance.

```
(config-Snmp=1)>dn ManagedElement=NODE06ST, System  
Functions=1, SecM=1  
(config-SecM=1)>
```

- By entering command **dn**, followed by **-m** or **-moc**, and a MOC, to navigate to the MOC instance. This is only applicable when there is one instance. If there are several MOC instances, the navigation can be qualified further by using parameter **-c** or **-condition**.

```
(config-Snmp=1)>dn -m Fm ManagedElement=NODE06ST, S  
ystemFunctions=1, Fm=1  
(Fm=1)>  
  
>dn -m Schema -c schemaId=CmwPm =>  
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, =>  
Schema=CmwPm  
(Schema=CmwPm)>
```

- By entering command **back**, to navigate back to the previous location in the MO tree.

```
(config-SecM=1)>back  
(config-Snmp=1)>
```

- By typing **..** or entering command **up**, to return to the superior MO in the tree. For example, to change the current location to the parent MO instance:

```
(config-SystemFunctions=1)>up  
(config-ManagedElement=NODE06ST)>
```

The **..** element can be part of the RDN navigation command:

```
(config-SysM=1)>.., Fm=1  
(config-Fm=1)>
```

- By entering command **top**, to change the current location to the root position:

```
(config-SystemFunctions=1)>top  
(config)>
```

- By entering command **show-dn**, to display the LDN of the current location in the MO tree.

```
(config-Fm=1)>show-dn ManagedElement=NODE06ST, Sys  
temFunctions=1, Fm=1
```



- By entering command **prompt**, to display the current location continuously.

```
(config-Fm=1)>prompt $dn ManagedElement=N0DE06ST,S
systemFunctions=1,Fm=1>
```

3.4.1 Navigation Errors

The navigation error messages are described in Table 23.

Table 23 Navigation Error Messages

Error Message	Description
ERROR: Navigation history is empty	It is not possible to navigate back, as the navigation history is empty.
ERROR: Cannot navigate back to: <MO_LDN>	It is no longer possible to navigate back to the MO instance, as, for example, the MO has been deleted or the access rules have been changed.
ERROR: Multiple MO instances found: <LDN1> <LDN2>	Command dn is used with parameter -m or -moc and there are several instances of the specified MOC. The LDNs are included in the message.
ERROR: No instance found	Command dn is used with parameter -m or -moc and there is no instance of the specified MOC that the user can navigate to.

3.5 Change MO Attributes

MO attributes can be changed in Config mode only. If the changes are entered without error (that is, no printout is provided), the change is added to the transaction and the changed MO is locked. The changes are applied after a successful commit of the transaction. The lock is released by transaction **abort** or **commit**, as initiated by CLI commands or session time-out.

This section describes how to change attributes of the following data types:

- Single-valued attribute, see Section 3.5.1 Change Single-Valued Attribute on page 80
- Single-valued attribute value deletion, see Section 3.5.2 Delete Value of Single-Valued Attribute on page 82
- Sequence, see Section 3.5.3 Change Attribute Defined as Sequence on page 82
- Struct, see Section 3.5.4 Change Attribute Defined as Struct on page 94
- Passwords, see Section 3.5.5 Change Attribute Defined as Password or Passphrase String on page 95



- MO reference, see Section 3.5.6 Change Attribute Defined as MO Reference on page 97

3.5.1 Change Single-Valued Attribute

Command [`<path>`,] `<attribute_name>=<attribute_value>` changes a single-valued attribute. If no printout is displayed as a result, the operation is verified against the data type specific rules defined in the model. Then the attribute change is added to the transaction, and the changed MO is locked. An error printout is displayed if the operation fails, examples are provided in Table 24.

The changes are applied after command `commit` and the locks are released on success.

The input syntax for the attribute data types is identical to the printout syntax, as described in Section 3.3.1 Display Single-Valued Attribute on page 49, with the following exceptions:

- Strings can be entered with or without double quotation marks ("). A string including characters with a special meaning in the CLI ([, comma, =, [], ", and]) must be entered with quotation marks.
- Booleans are interpreted in a case-insensitive way (for example, "TRUE", "true", and "True") and are displayed in lower case.

Attributes defined as `EcimEmpty` cannot have any value, it conveys information by its presence or absence.

Table 24 Attribute Value Change Errors

Error Message	Semantics	Data Type
ERROR: Attribute not writable	The user has read permission but not write permission.	Any data type
ERROR: Invalid value ' <code><attribute_value></code> ' for attribute ' <code><attribute_name></code> '. Valid values are in range : [<code><min></code> , <code><max></code>]	The attribute value is out of range. The allowed values are in the specified range.	int8, int16, int32, int64, uint8, uint16, uint32, uint64
ERROR: Invalid value ' <code><value></code> ' for attribute ' <code><attribute_name></code> '. <code><additional information></code>	The string attribute value is incorrect, as it does not comply with a model validation rule. The additional information contains a description associated with the model validation rule.	String



Table 24 Attribute Value Change Errors

Error Message	Semantics	Data Type
ERROR: Invalid value '<value>' for attribute '<attribute_name>'. Valid values are strings in a specified format. Type: '<attribute_name>?' for more information on the format to use	The string attribute value is incorrect, as it does not comply with a model validation rule. This error message is displayed when no description is associated with the model validation rule. Information about the correct format can be retrieved by requesting verbose help on the attribute.	String
ERROR: Invalid value '<stringattribute_value>' for attribute '<attribute_name>'. This is not a valid escape sequence	The attribute value is incorrect, as has an invalid escape sequence. The supported escape sequences are provided in Table 13.	String
Invalid value "<attribute_value>" for attribute "<attribute_name>". Valid values are: true, false	The attribute value is incorrect. The allowed values are true and false .	Boolean
ERROR: Invalid value '<structmember_value>' for struct member '<structmember_name>'. This is not a valid escape sequence	The struct member value is incorrect, as it has an invalid escape sequence. The supported escape sequences are provided in Table 13.	String
ERROR: Invalid value '<attribute_value>' for parameter '<attribute_name>'	Invalid command syntax.	String

```
(config-ManagedElement=NODE06ST)>dnPrefix=test
(config-ManagedElement=NODE06ST)>commit
```

Example 102 Change Single-Valued Attribute for String

```
(config-agentAddress)>port=xyz
ERROR: Invalid value 'xyz' for attribute 'port'.
Valid values are in range : [0,4294967295]
```

Example 103 Change Single-Valued Attribute for Integer

```
(config-Snmp=1)>operationalState=DISABLED
```

Example 104 Change Single-Valued Attribute for Enumeration



```
(config-agentAddress)>host="$4546"
```

Example 105 Change Single-Valued Attribute

```
(config-ManagedElement=NODE06ST)>myEmptyAttribute  
(config-ManagedElement=NODE06ST)>commit
```

Example 106 Change Single-Valued Attribute for EcimEmpty

3.5.2 Delete Value of Single-Valued Attribute

Command **no <nillable_attribute_name>** deletes an attribute value.

For examples on related verification rules and errors, see Table 25.

Table 25 Attribute Value Delete Errors

Error Message	Semantics
ERROR: Attribute <attribute_name> is read-only (can't be deleted)	A read-only attribute cannot be deleted.
ERROR: Delete only works for attribute and object types, type unrecognized for ("<unknownType>")	A delete operation can be performed only on attributes and object types.

```
(config-ManagedElement=NODE06ST show userLabel  
userLabel=xyz  
(config-ManagedElement=NODE06STcommit -s  
(config-ManagedElement=NODE06ST show userLabel  
userLabel=xyz  
(config-ManagedElement=NODE06STno userLabel  
(config-ManagedElement=NODE06STcommit -s  
(config-ManagedElement=NODE06ST show userLabel  
userLabel=xyz
```

Example 107 Delete Value of Single-Valued Attribute

```
(config-Snmp=1)>no operationalState
```

Example 108 Delete Value of Single-Valued Attribute

3.5.3 Change Attribute Defined as Sequence

This section describes how to change a sequence.



3.5.3.1 Initialize Sequence

Command `<sequence_name>=[<sequence_element>,<sequence_element>,...]` initializes a sequence.

```
(config-PlainMultivalueAttrs=1)>ddtStringMultivalueAttr=[ALABAMA,ALASKA,ARIZONA]
(config-PlainMultivalueAttrs=1)>commit -s
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
-----
--
--
--
--
```

Example 109 Initialize Sequence

3.5.3.2 Add Simple Type Element to Sequence

Command `<attribute_name>=<sequence_element_value>` adds an element to the last position of a sequence.

```
(config-PlainMultivalueAttrs=1)>ddtStringMultivalueAttr=[ALABAMA,ALASKA,ARIZONA]
(config-PlainMultivalueAttrs=1)>ddtStringMultivalueAttr=FLORIDA
(config-PlainMultivalueAttrs=1)>commit -s
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
-----
--
--
--
--
```

Example 110 Add Single-Valued Element to Sequence

3.5.3.3 Insert Simple Type Elements to Sequence

Command `insert [<path>,<attribute_name>[<existing_sequence_element>]=<new_sequence_element>` inserts a sequence element before the selected value of a sequence.



```
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
ddtStringMultivalueAttr
  "ALABAMA"
  "ALASKA"
  "ARIZONA"
(config-PlainMultivalueAttrs=1)>insert ddtStringMultivalueAttr⇒
[ALASKA]=FLORIDA
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
-----
-
-
-
-
```

Example 111 Insert Simple Type Element to Sequence

3.5.3.4 Insert Simple Type Element in Sequence Using Index

Command `insert [<path>,] <attribute_name>['@<index>']='<new_value>` inserts a sequence element into a sequence with existing values. The positional index starts from 1. The index must address an existing position in the sequence. Values cannot be added after the last existing value.

```
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
intMultivalueAttr
  42
  43
(config-PlainMultivalueAttr=1)>insert intMultivalueAttr[@1]=34
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
-----
-
-
-
```

Example 112 Insert Simple Type Element in Sequence Using Index

For examples on related verification rules and errors, see Table 26 and Table 29.

Table 26 Insert Simple Type Element in Sequence Using Index Errors

Error Message	Semantics
ERROR: Invalid index '<invalid_index>' to access a sequence containing <no_of_set_values> values. Use an index <=<no_of_set_values>	The index value is too large.
ERROR: Invalid index '<invalid_index>' to access a sequence containing 1 values. Use an index = 1	An index value greater than 1 is used to address a non-sequence attribute.



3.5.3.5 Change Sequence Element

Command [**<path>**,] **<attribute_name>**['**<existing_sequence_element>**']=**<new_sequence_element_value>** changes a sequence element.

```
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
ddtStringMultivalueAttr
    "ALASKA"
    "ALABAMA"
    "ARIZONA"
(config-PlainMultivalueAttrs=1)>ddtStringMultivalueAttr[ALABAMA]=⇒
FLORIDA
(config-PlainMultivalueAttrs=1)>commit -s
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
```

Example 113 Change Sequence Element

3.5.3.6 Change or Add Sequence Element Using Index

Command [**<path>**,] **<attribute_name>**['**@<index>**']=**<new_value>** changes or adds a sequence element. The positional index starts from 1. If the index addresses an existing sequence element, that value is replaced. If the index is one greater than the current number of values, the new value is added to the sequence.

```
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
intMultivalueAttr
    42
    43
(config-PlainMultivalueAttr=1)>intMultivalueAttr[@1]=34
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
```

Example 114 Change Sequence Element Using Index



```
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
intMultivalueAttr
  42
  43
(config-PlainMultivalueAttr=1)>intMultivalueAttr[@3]=34
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
```

Example 115 Add Sequence Element Using Index

For examples on related verification rules and errors, see Table 27 and Table 29.

Table 27 Change and Add Sequence Elements Using Index Errors

Error Message	Semantics
ERROR: Invalid index '<invalid_index>' to access a sequence containing <no_of_set_values> values. Use an index <= <no_of_set_values> for replace or use an index = <no_of_set_values+1> for append.	The index value is too large. It must address either an existing value, or the index after the last existing value. In the latter case, a value can be appended to the sequence.
ERROR: Invalid index '<invalid_index>' to access a sequence containing 1 value. Use an index = 1 for replace or use an index = 2 for append.	The index value is too large for an attribute having only one value. It must address either an existing value or the index after the last existing value. In the latter case, a value can be appended to the sequence.
ERROR: Invalid index '<invalid_index>' to access a sequence containing <no_of_set_values> values. Use an index <= <no_of_set_values> for replace.	The index value is too large. It must address an existing value. No more values can be appended to the sequence.
ERROR: Invalid index '<invalid_index>' to access a sequence containing 1 value. Use an index = 1 for replace.	An index value greater than 1 is used to address a non-sequence attribute.
ERROR: Invalid index '<invalid_index>' to access a sequence containing 0 values. Use an index = 1 for append.	An index value greater than 1 is used to address an attribute which has no values assigned to it.

To change a sequence attribute of type string when the value to be changed starts with character @, put the value within quotes to address it by value rather than by index.



```
(config-AMoc=1)>show userLabel
userLabel
  "AB"
  "@1"
(config-AMoc=1)>userLabel["@1"]=Swift
(config-AMoc=1)>show userLabel
```

Example 116 Change Sequence Attribute of Type String when Value Starts with @

3.5.3.7 Delete Named Element from Sequence

Command **no <attribute_name>=<sequence_element_value>** deletes a named element from a sequence.

```
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
ddtStringMultivalueAttr
  "ALASKA"
  "ALABAMA"
  "ARIZONA"
(config-PlainMultivalueAttrs=1)>no ddtStringMultivalueAttr=ALABAMA
(config-PlainMultivalueAttrs=1)>commit -s
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
```

Example 117 Delete Named Element from Sequence

3.5.3.8 Delete Element in Sequence Using Index

Command **no [<path>,] <attribute_name>'["@<index>"]'** deletes a sequence element in sequence. The positional index starts from 1. The index must address an existing position in the sequence.

```
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
intMultivalueAttr
  42
  43
(config-PlainMultivalueAttr=1)>no intMultivalueAttr[@1]
(config-PlainMultivalueAttr=1)>show intMultivalueAttr
intMultivalueAttr
  43
```

Example 118 Delete Element from Sequence Using Index

For examples on related verification rules and errors, see and Table 29.



Table 28 Delete Element from Sequence Using Index Errors

Error Message	Semantics
ERROR: Invalid index '<invalid_index>' to access a sequence containing <no_of_set_values> values. Use an index <= <no_of_set_values>	The index value is too large. It must address an existing value.
ERROR: Invalid index '<invalid_index>' to access a sequence containing 1 values. Use an index = 1	The index value is too large. It must address an existing value for an attribute that contains only one value.

3.5.3.9 Delete All Elements from Sequence

Command `no <sequence_name>` deleted all elements from a sequence.

Note: Deleting all elements from a sequence with this command results in an empty attribute. If an attribute defined as a sequence is instead initialized to the empty sequence by `<attribute_name>=[]`, it is guaranteed to be empty.

```
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
ddtStringMultivalueAttr
  "ALASKA"
  "ALABAMA"
  "ARIZONA"
(config-PlainMultivalueAttrs=1)>no ddtStringMultivalueAttr
(config-PlainMultivalueAttrs=1)>commit -s
(config-PlainMultivalueAttrs=1)>show ddtStringMultivalueAttr
```

Example 119 Delete All Elements from Sequence

3.5.3.10 Change Element in Sequence of Struct without Key

If an attribute is defined as a sequence of struct and the struct has no key member, a sequence element can be identified using the positional index. Thus, an element in the sequence can be modified by addressing that element with the positional index. The positional index is supported only for sequence of keyless structs. The positional index is identified by character `@` for all the struct elements in the sequence. The index numbering starts from 1.



```
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
  host="1.1.1.1"
  port=26343
agentAddress[@2]
  host="2.2.2.2"
  port=23154
(config-Snmp=1)>agentAddress[@2],host=4.4.4.4 port=4444
(config-Snmp=1)>show -v agentAddress
.....
.....
.....
.....
.....
.....
.....
```

Example 120 Change Element in Sequence of Struct without Key

3.5.3.11 Add a New Keyless Struct Element to Sequence

A new element can be appended to the end of the sequence in the following ways:

- Without specifying the positional index:

```
[<path> ,] <noKeyStructSequence>
```

- Setting member without the specifying positional index:

```
[<path> ,] <noKeyStructSequence> ,member_Name=<member_value>
```

- Specifying the positional index:

```
[<path> ,] <noKeyStructSequence>[@<N>]
```

- Setting member by specifying the positional index:

```
[<path> ,] <noKeyStructSequence>[@<N>] ,member_Name=<member_value>
```

A new struct instance is added to the end of the sequence. The user navigates to the newly created instance if the multiplicity for the attribute has not reached its maximum value.

If the struct members are set as part of the append operation, the specified values are set to the struct members.



```
(config-Snmp=1)>agentAddress
up
(config-Snmp=1)>show -v agentAddress
```

Example 121 Add a New Keyless Struct Element to Sequence without Index

```
(config-Snmp=1)>agentAddress[@2]
up
(config-Snmp=1)>show -v agentAddress
```

Example 122 Add a New Keyless Struct Element to Sequence with Index

```
(config-Snmp=1)>agentAddress,port=999
(config-Snmp=1)>show agentAddress
```

Example 123 Add a New Struct Element to Sequence by Setting Struct Member, without Index

```
(config-Snmp=1)>agentAddress[@3],host=9.9.9.9
(config-Snmp=1)>show agentAddress
```

Example 124 Add a New Struct Element to Sequence by Setting Struct Member, with Index

```
(config-Snmp=1)>agentAddress[@5]
```

Example 125 Add a New Keyless Struct Element to Sequence with Invalid Positional Index



3.5.3.12 Delete Keyless Struct Sequence

Command **no [<path> ,] <noKeyStructSequence>[@<N>]** deletes an element at position N in a sequence of keyless structs. All instances after position N are shifted up by one position.

Command **no <noKeyStructSequence>** deletes all struct elements in the sequence.

```
(config-Snmp=1)>show -v agentAddress
```

```
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
```

```
(config-Snmp=1)>no agentAddress[@2]
(config-Snmp=1)>show -v agentAddress
```

```
-----
-----
-----
-----
-----
-----
-----
```

Example 126 Delete Keyless Struct Element at Positional Index <N>

```
(config-Snmp=1)>show -v agentAddress
```

```
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
```

```
(config-Snmp=1)>no agentAddress[@2],host
(config-Snmp=1)>show -v agentAddress
```

```
-----
-----
-----
-----
-----
```

Example 127 Delete Struct Member of Struct Element in Sequence of Keyless Struct



```
(config-Snmp=1)>show agentAddress
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
(config-Snmp=1)>no agentAddress
```

```
(config-Snmp=1)>show agentAddress
```

```
agentAddress
```

Example 128 Delete Whole Sequence of Keyless Struct

3.5.3.13

Insert Struct Element to a Sequence of Keyless Structs

Command **insert [<path>,] <noKeyStructSequence>[@<N>]** inserts a struct element at position N to a sequence of keyless structs. The struct elements after position N are moved down one position.

```
(config-Snmp=1)>insert agentAddress[@2]
```

```
(config-agentAddress[@2])>host=9.9.9.9
```

```
(config-agentAddress[@2])>port=9999
```

```
(config-agentAddress[@2])>up
```

```
(config-Snmp=1)>show -v agentAddress
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
agentAddress
```

```
192.168.1.1
```

```
192.168.1.1
```

```
agentAddress
```

Example 129 Insert Element to Sequence of Keyless Struct

3.5.3.14

Change Element in Sequence of Keyed Struct

If the attribute is defined as a sequence of struct and the struct has a key member, a sequence element can be identified and the CLI position can be changed in the sequence.

As a result, existing sequence elements can be changed and new elements can be added without changing the existing element, as shown in Example 130.



```
(config-PlainMultivalueAttrs=1)>countries=Sweden
(config-countries=Sweden)>capital=Stockholm
(config-countries=Sweden)>population=2000000
(config-countries=Sweden)>up
(config-PlainMultivalueAttrs=1)>countries=Norway
(config-countries=Norway)>capital=Oslo
(config-countries=Norway)>up
(config-PlainMultivalueAttrs=1)>show countries

(config-PlainMultivalueAttrs=1)>show countries=Sweden

(config-PlainMultivalueAttrs=1)>no countries=Sweden
(config-PlainMultivalueAttrs=1)>show countries
```

Example 130 Change Element in Sequence of Keyed Struct

The country attribute of class PlainMultivalueAttrs is defined as a sequence of struct. The struct has the following members:

- “name”, defined as string, that is, the key of the struct
- “capital”, defined as a string
- population, defined as int64 with default value 0

3.5.3.15 Common Error Messages in Sequence Operations

The common error messages in sequence operations are shown in Table 29.

Table 29 Common Error Messages in Sequence Operations

Error Message	Semantics
ERROR: Value must be unique	The sequence elements are set to be unique, that is, property nonUnique is not present.
ERROR: Multiplicity of the attribute (" <attribute_name> ") at max limit	The sequence element property maxLength is set and the number of elements is higher than this limit.



Table 29 Common Error Messages in Sequence Operations

Error Message	Semantics
ERROR: Multiplicity of the attribute (" <attribute_name> ") at minimum limit	The sequence element property minLength is set and the number of elements is lower than this limit.
ERROR: Invalid index <positional_Index> to access a sequence containing <existing number of values> values. Use an index <= <valid_values>	The keyless struct element at the specified positional index for a keyless struct sequence does not exist. The allowed values are less than or equal to <valid_values>.

3.5.4 Change Attribute Defined as Struct

Command `<struct_name,struct_element>=<struct_element_value>` changes a struct. If nothing is displayed as a result, the changed struct is added to the transaction and the changed MO is locked. An error printout is displayed if the operation fails, see Table 30.

If struct property `isExclusive` is set, setting one struct member means that all other struct members are automatically unset.

The changes are applied after command `commit` and the locks are released on success. Navigating away from the MO can only be done if all members of a struct inside the MO are set correctly.

Table 30 Common Error Message in Struct Operations

Error Message	Semantics
ERROR: Failed to set struct key attribute '<attribute_name>' to '<attribute_value>'. Key must be unique.	The key value is not unique.

```
(config-ManagedElement=NODE06ST)>productId=1,productDesignation=xyz
(config-ManagedElement=NODE06ST)>productId=1,productNumber=1234
(config-ManagedElement=NODE06ST)>productId=1,productRevision=1.1
(config-ManagedElement=NODE06ST)>commit -s
```

Example 131 Change Struct

3.5.4.1 Struct of Sequence

A struct can have a sequence as its element. Initialization, adding, and modifying values of such struct elements is similar, as described in Section 3.5.3 Change Attribute Defined as Sequence on page 82.



3.5.5 Change Attribute Defined as Password or Passphrase String

Two types of passwords/passphrases exist and they are treated slightly differently in the CLI. In this section, the terms legacy password and passphrase string are used to distinguish the types.

Legacy passwords can appear only as single-valued attributes, whereas passphrase strings can appear both as single valued and as sequences, and both as attributes and struct members. Verbose help shows a legacy password as type password and a passphrase string as a type string with tag passphrase.

The way to enter values of both types interactively depends on how the ME is configured. Either they are entered visibly, similar to how other attribute types are assigned, or they are entered hidden using special prompts. For details, see Section 3.5.5.1 Visible Entry of Values on page 95 and Section 3.5.5.2 Hidden Entry of Values on page 96.

When entered non-interactively, for example, in script mode or when commands are pasted into the CLI, the values are entered visibly.

3.5.5.1 Visible Entry of Values

A legacy password is entered in the following ways:

- An encrypted value can be assigned to a password attribute by `<attribute_name>=<attribute_value>`.
- A cleartext value can be assigned to a password attribute by `<attribute_name>=<attribute_value> cleartext`.

A passphrase string value is assigned by `<attribute_name>=<attribute_value>`, where `<attribute_value>` must be cleartext, unless the ME is configured to accept encrypted values. The information about whether a value is encrypted or not is in this case encoded within the value itself.

For an encrypted value to be valid, it must be taken from the output of an CLI `show` type command. An encrypted value is normally only valid on one ME, but MEs can be configured so that values can be copied between them. The encrypted values of legacy passwords and passphrase strings are only valid as input of the respective type, that is, encrypted values cannot be copied between the two types.

Cleartext values of both types become encrypted when they are entered. When displaying legacy passwords, the values are shown encrypted. When displaying passphrase strings, the values are normally not displayed, but an ME can be configured to display encrypted values.

```
(config-A=1)>myPassword="foobarmypasswd398" cleartext
(config-A=1)>show myPassword
```

Example 132 Non-Encrypted Legacy Password



```
(config-A=1)>myPassword="34sfsSFGargy5ghyj124"  
(config-A=1)>show myPassword
```

Example 133 Already Encrypted Legacy Password

```
(config-A=1)>myPassphrase="foobarmypasswd398"  
(config-A=1)>show myPassphrase
```

Example 134 Passphrase String, Encrypted Input and Output is Not Allowed

```
(config-A=1)>myPassphrase="foobarmypasswd398"  
(config-A=1)>show myPassphrase  
myPassphrase="encrypted:hdg4jdldf8gfk5jd"  
(config-A=1)>myPassphrase="encrypted:hdg4jdldf8gfk5jd"  
(config-A=1)>show myPassphrase
```

Example 135 Passphrase String, Encrypted Input and Output is Allowed

3.5.5.2

Hidden Entry of Values

When interactively entering a legacy password or a passphrase string, the following applies:

- In the following cases, the cursor moves to a new line, where a special prompt is displayed for the entry of the value:
 - When the equals sign (=) is entered after the name of an attribute or struct member
 - When the **Tab** key is used to complete the name of such attribute or struct member

The characters entered at this prompt are not echoed.

- When the **CR** or **Enter** key is pressed, a second prompt is displayed for the repeated entry of the value.

If the **CR** or **Enter** key is pressed at this prompt, and if the two entered strings are identical, the normal command line is displayed again. A sequence of eight asterisks is representing the password, see Example 136.

```
(config-A=1)>myPassword=  
  
(config-A=1)>myPassword=*****
```

Example 136 Entry of Hidden Legacy Password or Passphrase String

The contents of the command line where the value is represented by asterisks cannot be edited.



If the type is legacy password, two alternatives are available:

- To change the attribute member, press the **CR** key or the **Enter** key.
- To clear the line, press **Ctrl+C**.

If the type is passphrase string, further attributes or struct members can be changed on the same line.

If the two entered values are not identical, an error message is displayed and the first password prompt is redisplayed, as shown in Example 137.

```
(config-A=1)>myPassword=
<password>
<another_password>
```

Example 137 Mismatching Passwords

To cancel the entry of the password strings, press **Ctrl+C**.

Note: An encrypted password cannot be entered in this way. The entered values are automatically treated as cleartext. If a legacy password attribute must be changed to an already encrypted value, this is done either in script mode or by pasting a full line including the encrypted value into the CLI. However, encrypted passphrase string values can be entered at these special prompts.

3.5.6 Change Attribute Defined as MO Reference

Changing an attribute of MO reference type is the same as changing a string attribute with the following special conditions:

- The MO reference attribute value is the LDN of the referred MO.
- If the definition of the MO reference attribute contains a certain MOC type, the referred MO must be an instance of this MOC.
- If the definition of the MO reference attribute has external Unidirectional Association, the referred MO accepts any arbitrary string.
- The referred MO instance does not have to exist. If the ManagedElement ID does not match the one defined in the system, the CLI automatically corrects it.

As an alternative to providing the full LDN, the CLI also allows the value to be an MO path relative to the current position, using the same rules as when navigating to an MO instance. This makes it possible to, for example, navigate to the MO instance that is to be referred to, and then give the MO reference attribute the value “.”. When assigning the value, the CLI automatically translates the relative path to an LDN.

An MO reference value can, as any string, be entered unquoted, as long as it does not contain any space characters. If an MO instance name contains a space, only the MO instance name can be quoted within the DN. However, it is recommended to quote the full MO reference values, as syntactical ambiguities can arise, in particular in commands changing sequences of MO references.

3.5.7 Change Multiple Attributes on One Command Line

Multiple attributes belonging to the same MO instance can be changed on a single command line, by separating the subcommands with space characters.

The supported modifications are as follows:

- Change single-value attribute, see Section 3.5.1 Change Single-Valued Attribute on page 80.
- Initialize sequence, see Section 3.5.3.1 Initialize Sequence on page 83.
- Add single-value element to sequence, see Section 3.5.3.2 Add Simple Type Element to Sequence on page 83.

The CLI handles all subcommands on the same command line as one atomic command. That is, the result is either that all the subcommands on the command line are executed successfully or no changes are made.

```
(config-MocA=1)>intMultiValueAttr=1 stringAttr="abc" =>
enumMultiValueAttr=[ONE, TWO]
(config-MocA=1)>show
```

Example 138 Change Multiple Attributes on One Command Line

To change attributes outside the current MO location, give the path to the MO where the attributes are located, followed by a comma (,), and then list the attributes changes.

```
(config-MocC=1)>MocB=1,MocA=1,intMultiValueAttr=1 =>
stringAttr="abc" enumMultiValueAttr=[ONE, TWO]
(config-MocC=1)show MocB=1,MocA=1
```

Example 139 Change Multiple Attributes on One Command Line from Outside MO



Multiple struct members can also be set on one command line according to the principles that apply for attributes.

3.6 Create MO

MOs can be created in an atomic way in a transaction, that is, in Config mode only. As a consequence, the configuration changes are not applied by entering the changes, but after successful **commit** of the transaction.

To create an MO:

1. Enter Config mode:

```
>configure
```

2. Check if the MO exists:

```
(config)>show <path>
```

The CLI command for MO creation is identical to the CLI command for changing the CLI position to an existing MO. The successes of both operations are indicated in the same way, by providing no printout.

3. Select the appropriate action according to the possible results:
 - No error message is displayed, but the requested MO is displayed. In this case, the MO exists. Continue by changing the MO attributes, as described in Section 3.5 Change MO Attributes on page 79, according to the required changes.
 - Error message `ERROR: Specific element not found` indicates that the MO does not exist. Continue with the next step.
4. Enter the name of the MO according to the MO naming rules (see Section 3.6.1 MO Naming Rules on page 101) to create the desired MO.

Example of key attribute of type string:

```
(config)>ManagedElement=N0DE06ST, SystemFunctions=1, ⇒  
SysM=1, NtpServer=myServer
```

If the MO key attribute is of enumeration type, press the **Tab** key to list the available values with information about which values that are not yet instantiated, for example:

```
(config-KeyAttrMOC=1)>EnumKeyAttrMOC=<Tab>  
ONE  
THREE <new>  
TWO <new>
```



The possible results are as follows:

- If no error printout is displayed, the operation succeeded and the CLI position changes to the new MO. The parent of the newly created MO, and the MO itself, is locked, for example, (config-NtpServer=myServer)>.

Note: If the MO is created and the value of a mandatory attribute is not assigned, no error will be thrown and the CLI position will be changed. Mandatory and cardinality related constraints can be checked by issuing a validate or commit command.

- An error message is displayed if verification fails against any available constraint. In this case, modify the DN to comply with the constraints. For examples of error messages, see Table 31.

5. Press the **Tab** key to list the available optional and mandatory attributes, for example:

```
(config-NtpServer=AServer)><Tab>
administrativeState
serverAddress
userLabel
```

6. Request Verbose Help to determine if the attribute is mandatory and if it has a default value, for example:

```
(config-NtpServer=AServer)>administrativeState ?
administrativeState BasicAdmState <LOCKED|UNLOCKED>
[optional]
Locks or unlocks the operation of the NTP client
function.
This is a convenience function to permit some or all
NtpServer instances to be temporarily locked without
having to delete the object.
```

7. Set the mandatory attributes and the needed optional attributes, for example:

```
(config-NtpServer=myServer)>serverAddress="22.22.22.22"
```

8. Commit the creation:

```
(config-NtpServer=myServer)>commit
```

9. Verify the result.

The possible results are as follows:

- If no result message is displayed, the MO creation succeeded. In this case, the CLI position is in the new MO position in Config mode and a new transaction is created automatically. The lock on the parent of the newly created MO is released.



- If the MO creation failed, the error message is displayed on the error reason. In this case, act according to the error indication.

Table 31 MO Creation Errors

Error Message	Semantics
ERROR: Parent '<parent_DN>' does not exist	The parent DN does not exist.
ERROR: Can not instantiate system created object	Instantiation of system-created objects is not allowed.
ERROR: Cardinality is at upper limit for class (<MO_class_name>), cannot create object	An MO instance cannot be created because of a restriction on cardinality.
ERROR: MO creation failed for classname: <MO_class_name>, error code: <error_code>	Other implementation-specific cases.
ERROR: No create permission for '<DN>'	The user has read permission but not create permission.
ERROR: Command not found	The user has not read or write permission.
ERROR: Instances of '<moc_name>' are not creatable	The MO cannot be created.

```

(config-Snmp=1)>SnmpTargetV1=OSS-42
(config-SnmpTargetV1=OSS-42)>_
...
...
...
...
...
(config-SnmpTargetV1=OSS-42)>address=1.2.3.4
(config-SnmpTargetV1=OSS-42)>administrativeState=UNLOCKED
(config-SnmpTargetV1=OSS-42)>community=zoneA
(config-SnmpTargetV1=OSS-42)>isMibWritable=true
(config-SnmpTargetV1=OSS-42)>port=777
(config-SnmpTargetV1=OSS-42)>commit

```

Example 140 Create MO

3.6.1 MO Naming Rules

The MO naming rules depend on the type of the key attribute.



3.6.1.1 Strings

The MO naming rules for strings are as follows:

- All ASCII characters in range 32–126 are supported except the following restricted characters:
 - Comma (,)
 - Equals sign (=)
 - Plus sign (+)
 - Less-than sign (<)
 - Greater-than sign (>)
 - Number sign (#)
 - Semicolon (;)
 - Reverse slash (\)
 - Quotation mark (")
 - Asterisk (*)
- All other characters, including the restricted characters, must be entered as \XX, where `XX` is the two-digit hexadecimal ASCII code of the character.

3.6.1.2 Integers

For integers, the value must be numeric and inside the range of the underlying type, which can be `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, or `uint64`.

3.6.1.3 Enumerations

For enumerations, the value be the name of a member of an enumeration.

3.7 Reinitialize MO

Command `reinit [<path>,]<attribute_name>` resets either of the following to its initial state:

- An MO instance
- A single attribute
- A single struct member
- A struct instance within a sequence of structs



`<attribute_name>` is either an attribute, a struct, or a struct instance within a sequence of structs based on key or index and struct member.

This command can be used in Config mode only.

The initial state of an MO instance is what it has when all its attributes have their default values, or (if they lack default values) no values. Command `reinit` only affects one single MO instance, it does not reinitialize or delete any child MO instances.

The initial state of an attribute is what it has when an MO instance is created. That is, if an attribute has a default value, command `reinit` sets its value to the default value, otherwise to empty.

Read-only attributes cannot be reinitialized, as their values are assigned by the system. Key attributes and key struct members cannot be reinitialized either. Restricted attributes can be reinitialized only in newly created and not yet committed MO instances. Such non-resettable attributes and struct members are ignored when a whole MO or a struct instance is reinitialized.

Note: When an attribute, that is, a sequence of structs, is reinitialized, all existing struct instances are deleted, and if the minimum multiplicity is higher than zero, a new struct instance is created. To keep and reinitialize all existing struct instances, the instances must be reinitialized one by one.

When an MO instance or an attribute is reinitialized, the minimum multiplicity constraints of the attribute cannot be fulfilled. In this case, a warning message is displayed and the transaction cannot be committed until the condition is corrected. The warning messages have the format
 WARNING: Incomplete object (<path>)
 Minimum multiplicity of <min> is violated for attribute(<attribute_name>).

Table 32 MO Reinitialize Errors

Error Message	Semantics
ERROR: Cannot reinitialize key attribute <attribute_name>	A key attribute cannot be reinitialized.
ERROR: Cannot reinitialize read-only attribute <attribute_name>	A read-only attribute cannot be reinitialized.
ERROR: Cannot reinitialize restricted attribute <attribute_name>	A restricted attribute cannot be reinitialized.
ERROR: Cannot reinitialize the key member <struct_member_name> of the struct <struct_id>	A key member of a struct cannot be reinitialized.



Table 32 MO Reinitialize Errors

Error Message	Semantics
ERROR: <attribute_name> does not have any instances	A struct that does not have any instance cannot be reinitialized.
ERROR: Cannot reinitialize attribute, no write permission for <attribute_name>	An attribute that does not have write permission cannot be reinitialized.

Examples

In Example 141 attribute `attrWithDefaultValue` has default value "abc" and attribute `attrWithoutDefaultValue` has been assigned value 45. Command **reinit** resets the value of attribute `attrWithDefaultValue` to empty.

```
(config-TestRootMoc=1)>show -v
TestRootMoc=1
  attrWithDefaultValue="abc" <default>
  attrWithoutDefaultValue=45
  aSimpleStruct
    memberWithDefaultValue=123 <default>
    memberWithoutDefaultValue=23
(config-TestRootMoc=1)>reinit
(config-TestRootMoc=1)>show -v
```

Example 141 Reset MO

In Example 142 attribute `attrWithDefaultValue` has default value "abc" and the attribute is assigned value 123. Command **reinit** resets attribute `attrWithDefaultValue` to its default value.

```
(config-TestRootMoc=1)>show -v attrWithDefaultValue
attrWithDefaultValue="abc" <default>
(config-TestRootMoc=1)>attrWithDefaultValue=123
(config-TestRootMoc=1)>show -v attrWithDefaultValue
attrWithDefaultValue="123"
(config-TestRootMoc=1)>reinit attrWithDefaultValue
(config-TestRootMoc=1)>show -v attrWithDefaultValue
```

Example 142 Reset Attribute

In Example 143 attribute `mandatoryAttr` has minimum multiplicity 1 and no default value. Command **reinit** resets the attribute value and a warning message is displayed on violation of minimum multiplicity constraints.



```
(config-TestRootMoc=1)>show -v
TestRootMoc=1
  mandatoryAttr=123
(config-TestRootMoc=1)>reinit mandatoryAttr
WARNING: Incomplete object (ManagedElement=NODE06ST,TestRootMoc=1)
Minimum multiplicity of 1 is violated for attribute (mandatoryAttr)
(config-TestRootMoc=1)>show -v mandatoryAttr
```

Example 143 Reset Command with Minimum Multiplicity Constraints Violation

Command **reinit** deletes all instances in a sequence of structs and creates one instance when only the struct attribute name is given as parameter, see Example 144. Otherwise, when a struct key value or an index is given, only the members of the specified instance are reinitialized, excluding any key member, see Example 145 and Example 146.

```
(config-PlainMultivalueAttrs=1)>show -v aKeylessStruct
aKeylessStruct[01]
  int1=[] <empty>
  str1=[] <empty>
aKeylessStruct[02]
  int1=3
  str1=[] <empty>
(config-PlainMultivalueAttrs=1)>reinit aKeylessStruct
WARNING: Incomplete object (ManagedElement=NODE06ST,TestRootMoc=1,MultivalueThing=1,PlainMu
Minimum multiplicity of 1 is violated for attribute (aKeylessStruct) struct member: str1
Minimum multiplicity of 1 is violated for attribute (aKeylessStruct) struct member: int1
(config-PlainMultivalueAttrs=1)>show -v aKeylessStruct
aKeylessStruct[01]
  int1=[] <empty>
  str1=[] <empty>
```

Example 144 Reset struct

```
(config-ResetTestMocWithMultiValueAttr=1)>show -v mandatoryKeyedStruct
mandatoryKeyedStruct
  capital=[] <empty>
  name="1" <key>
  population=0 <default>
mandatoryKeyedStruct="2"
  capital="1"
  name="2" <key>
  population=0 <default>
(config-ResetTestMocWithMultiValueAttr=1)>reinit mandatoryKeyedStruct=2
(config-ResetTestMocWithMultiValueAttr=1)>show -v mandatoryKeyedStruct
mandatoryKeyedStruct
  capital=[] <empty>
  name="1" <key>
  population=0 <default>
mandatoryKeyedStruct="2"
  capital=[] <empty>
  name="2" <key>
  population=0 <default>
```

Example 145 Reset a Specific Instance of a struct with a Key Member



```
(config-ResetTestMocWithMultiValueAttr=1)>show -v sequenceOfKeylessStruct
sequenceOfKeylessStruct[@1]
  int1=3
  str1="3"
(config-ResetTestMocWithMultiValueAttr=1)>reinit =>
sequenceOfKeylessStruct[@1]
(config-ResetTestMocWithMultiValueAttr=1)>show -v =>
sequenceOfKeylessStruct
```

Example 146 Reset a Specific Instance of a struct without a Key Member

In Example 147 attribute `attrWithDefaultValue` has default value "abc" and attribute `attrWithoutDefaultValue` has been assigned value 45. Command `reinit` resets the value of attribute `attrWithDefaultValue` to empty.

```
(config-ManagedElement=NODE06ST)>show -v TestRootMoc=1
TestRootMoc=1
  attrWithDefaultValue="abc" <default>
  attrWithoutDefaultValue=45
  aSimpleStruct
    memberWithDefaultValue=123 <default>
    memberWithoutDefaultValue=23
(config-ManagedElement=NODE06ST)>reinit =>
ManagedElement=NODE06ST, TestRootMoc=1
(config-ManagedElement=NODE06ST)>show -v TestRootMoc=1
```

Example 147 Reset MO using LDN

In Example 148 attribute `attrWithDefaultValue` has default value "abc" and the attribute is assigned value 123. Command `reinit` resets attribute `attrWithDefaultValue` to its default value.

```
(config)>show -v ManagedElement=NODE06ST,TestRootMoc=1,attrWithDefaultValue
attrWithDefaultValue="abc" <default>
(config)>ManagedElement=NODE06ST,TestRootMoc=1,attrWithDefaultValue=123
(config)>show -v ManagedElement=NODE06ST,TestRootMoc=1,attrWithDefaultValue
attrWithDefaultValue="123"
(config-TestRootMoc=1)>reinit ManagedElement=NODE06ST,TestRootMoc=1,attrWithDefaultValue
(config-TestRootMoc=1)>show -v ManagedElement=NODE06ST,TestRootMoc=1,attrWithDefaultValue
attrWithDefaultValue="abc" <default>
```

Example 148 Reset Attribute with LDN

Command `reinit` deletes all instances in a sequence of structs and creates one instance when only the struct attribute name is given as parameter, as the minimum multiplicity is 1 for the struct, see Example 149.



```
(config-TestRootMoc=1)>show -v ManagedElement=NODE06ST,TestRootMoc=1,aKeylessStruct
aKeylessStruct[01]
  int1=[] <empty>
  str1=[] <empty>
aKeylessStruct[02]
  int1=3
  str1=[] <empty>
(config-TestRootMoc=1)>reinit ManagedElement=NODE06ST,TestRootMoc=1,aKeylessStruct
WARNING: Incomplete object (ManagedElement=NODE06ST,TestRootMoc=1)
Minimum multiplicity of 1 is violated for attribute (aKeylessStruct) struct member: str1
Minimum multiplicity of 1 is violated for attribute (aKeylessStruct) struct member: int1
(config-TestRootMoc=1)>show -v ManagedElement=NODE06ST,TestRootMoc=1,aKeylessStruct
aKeylessStruct[01]
  int1=[] <empty>
  str1=[] <empty>
```

Example 149 Reset Sequence of struct using LDN

3.8 Delete MO

MOs are deleted in Config mode only. The configuration changes are not applied by entering the changes, but after a successful commit of the transaction.

To delete an MO:

1. Enter Config mode:

```
>configure
```

2. Delete the MO by command **no** with the RDN of the MO, for example:

```
(config-Snmp=1)>no SnmpTargetV1=1
```

The possible results are as follows:

- If no result message is displayed, the MO deletion succeeded. In this case, the deleted MO, its parent MO, and its children MOs are locked.
- An error message is displayed if verification failed against any available constraint. In this case, modify the DN to comply with the constraints. For examples of error messages, see Table 33.

3. Commit the deletion:

```
(config-Snmp=1)>commit
```

4. Verify the deletion:

- If no result message is displayed, the MO deletion succeeded. In this case, the CLI position is not changed, the locks are released, and a new transaction is created automatically.
- If the MO deletion failed, an error message with error reason is displayed. In this case, act according to the error indication.



Note: An MO can be deleted even if it is pointed by an MO reference.

Table 33 MO Deletion Errors

Error Message	Semantics
ERROR: Cardinality is at lower limit for class (" <MO_class_name> "), cannot delete object	The object cannot be deleted because of a cardinality constraint.
ERROR: Can not delete system created object	Deletion of system-created object is not allowed.
ERROR: No delete permission for ' <DN> '	The user has read permission but not delete permission.
ERROR: Command not found	The user has not read or write permission.
ERROR: Instances of ' <MOC_name> ' are not deletable	The MO cannot be deleted.

3.9 MO Actions

Actions are executed in Config mode and Exec mode on existing MOs, but not actions that are created in a transaction and not yet committed (in the Config mode case).

An MO becomes locked in the following cases:

- If an attribute change is requested, the MO containing the attribute is locked.
- If an MO creation is requested, the MO and its parent MO are locked.
- If an MO deletion is requested, the MO, its parent MO, and its children MOs are locked.
- If an action execution is requested, the MO containing the action is locked.

The locks are released if the transaction is committed or aborted, or if the result of action execution is received.

3.9.1 Action Request

Command [**<path>**,] [**.,**]**<action_name>** [**--<action_parameter_name>** **<action_parameter_value>**] ... requests action execution.

Optional parameters are supported with this format. A parameter can be considered as optional if the parameter contains a default value in the model.

Values must be specified for all the parameters that are not optional. Action parameters defined as struct (for example, struct `EcimPassword`) or sequence multi-valued attribute are not supported.



If an action name is conflicting with any command names, give the action request as `'. ,<action_name>'` if the action must be executed from the current DN.

The syntax for executing an action, which has a name conflicting with the command name (`show`) from the current DN, is shown in Example 150.

```
(config-PrecedenceThing=2)> . , show
```

Example 150 Action Request

3.9.1.1 Alternative Hidden Password Entry

An ME can be configured to hide the characters of entered passwords. If that is the case, and script mode is off, the following applies:

- In the following cases, the cursor moves to a new line, where a special prompt is displayed for the entry of the password string:
 - When the **Space** key is pressed after the name of an action parameter defined as a password
 - When the **Tab** key is used to complete the name of such a parameter

The characters entered at this prompt are not echoed.

- When the **CR** key or the **Enter** key is pressed, the normal command line is presented again, and a sequence of eight asterisks represents the password, as shown in Example 151.

```
(Schema=1)>export --password
(Schema=1)>export --password *****
```

Example 151 Alternative Hidden Password Entry

The contents of the command line cannot be edited where the password is represented by asterisks, but more parameters can be added.

To cancel the entry of the password string, press **Ctrl+C**.

3.9.2 Response to Action Request

If the action request is completed with error, the error reason is displayed as `ERROR: <error_text>`. For examples on error messages, see Table 34. If an action request is completed without error, the following apply:

- No printout is provided if the action return value is defined as `void`.
- Otherwise, a printout is provided according to the action return type. The action printout format is identical to `show [<path> ,]<attribute>` of the same data type, see Section 3.3 Display Information on page 48.



Note: Error indication in an exception parameter is not supported.

Table 34 Action Error Messages

Error Message	Semantics
ERROR: No execute permission for action '<action_name>'	The user has no execute permissions but the action is visible.
ERROR: Too many arguments for action (<action_name>) that takes <number_of_parameters> parameters <paramname1, paramname2, ...>	The number of parameters is higher than expected.
ERROR: Too few arguments for action (<action_name>) that takes <number_of_parameters> parameters <paramname1, paramname2, ...>	The number of parameters is fewer than expected.
ERROR: Call command failed, error code <error_code>	Other implementation-specific cases.
ERROR: Invalid GNU Style Syntax for parameter : <param_name>	The parameter is not in GNU style.
ERROR: No parameter found with name : <param_name>	The parameter with the name specified is not found.
ERROR: Duplicate parameters not allowed : <param_name>	The parameter with the name has more than one occurrence in the command.
ERROR: No value found for parameter : <param_name>	No value is specified for the parameter name.
ERROR: Invalid value '<param_name>' for parameter '<paramname>'. Valid values are: ' LOCKED UNLOCKED SHUTTING_DOWN'. Type: '<action_name>?' for more information on the values to use.	Incorrect value is specified for the action parameter. By typing <action_name>? help information on the allowed values is displayed.
ERROR: Invalid value "<param_name>" for parameter "<paramname>". Valid values are: true, false	Incorrect value is specified for the action parameter. By typing <action_name>? help information on the allowed values is displayed.
ERROR: Invalid value '<param_name>' for parameter '<paramname>'. Valid values are in range : [<min>, <max>]	The value specified for the action parameter is out of range. The allowed values are in the specified range.
ERROR: Invalid value "<parameter_value>" for parameter "<parameter_name>". This is not a valid escape sequence	The parameter value is incorrect, as it does not have a valid escape sequence. The supported escape sequences are provided in Table 13.
ERROR: Invalid value "<parameter_value>" for parameter "<parameter_name>".	Invalid command syntax.



3.9.3 Action Start

Depending on the semantics, the success of an action request can mean that the action execution is completed or started, resulting in asynchronous or synchronous action execution.

The start of the action execution can be bound to various conditions depending on the action type such as the following:

- Immediate start – The action execution starts immediately after the request is entered.
- Start by commit – The action execution starts after the request is entered and the related transaction is committed.
- Confirmed start – To start the action execution, extra confirmation is needed.
- Start with timer – The action execution starts after a predefined time-out. This start can be combined with confirmed start.
- Any action-specific preconditions, for example, internal states and parallel activities.

For the action semantics and start conditions, see the action description in the Help description or in the model description.

Example: Immediate Action Start

1. Navigate to the MO where the action is present, for example:

```
>ManagedElement=NODE06ST,MOCex7=1,M0ex10=1
```

2. Trigger the action, for example:

```
(config-M0ex10=1)>addNumbers --num1 23 --num2 54
77
(config-M0ex10=1)>concatString_defValues
comuser
(config-M0ex10=1)>booleanAdder --flag2 True
false
```

Example: Action Start by Commit

1. Enter Config mode:

```
>configure
```

2. Navigate to the MO where the action is present, for example:

```
(config)>ManagedElement=NODE06ST,SystemFunctions=1,File
M=1,LogicalFs=1,FileGroup=SysMMimSchemas
```



3. Request the action execution, for example:

```
(config-FileGroup=SysMMimSchemas)>removeFile xyz.txt  
true
```

4. Trigger the action start:

```
(config-FileGroup=SysMMimSchemas)>commit
```

3.10 Copy and Paste Configuration Data

The CLI supports pasting of large lines of valid configuration data into the CLISS terminal. An unlimited number of configuration lines can be pasted. To make it possible to copy data from one node to another with a different name, the CLI automatically corrects the entered node names in all commands (ManagedElement ID).

To paste configuration data, copy a valid configuration from the command **show-config** output and paste it into the CLISS terminal.

Note: If any of the pasted input commands result in errors, the remaining pasted input cannot be successfully accepted. Also, this feature does not work as expected on some environments based on the terminal settings or the CPU speed.

3.11 Change Password Command

Command **passwd** changes the password of the logged on user.

This is an optional command. It is available only if the ME supports changing user passwords through the CLI. This command uses the built-in password changing utility of the ME, which means that the exact behavior depends on the ME type.

```
>passwd  
Changing password for cliuser.  
Old Password:<old_password>  
New Password:<new_password>  
Reenter New Password:<new_password>  
Password changed.
```

Example 152 Changing Password

Changing password can fail because of the reasons specified in Table 35.

Table 35 Change Password Error Messages

Error Message	Semantics
passwd: Authentication failure	The command is not entered in root position or the ME does not support changing user passwords through the CLI.



Table 35 Change Password Error Messages

Error Message	Semantics
Bad password: it is based on a dictionary word	Invalid password.
Bad password: too simple	Invalid password.
Bad password: too similar	Invalid password.
passwd: Have exhausted maximum number of retries for service	The user has retried to change the password too many times.

3.12 Display CLI Version

Command **version** displays the CLI version.

```
>version
```

Example 153 Display CLI Version

For more information, see Section 1.4 ECLI Version on page 3.

3.13 Pipe Utility Commands

Filtering of CLI and Subshell command output with the help of PIPE extension command modules is supported. The pipe symbol **|** is used to indicate that the output of the CLI and Subshell commands is sent as input for pipe extension commands.

3.13.1 Filter Command

The filter command (**<command> | <action> | filter [-i|--ignore] [-v|--invert] [{-A|--after} <value>] [{-B|--before} <value>] <pattern>**) is a pipe extension command to filter the output of any command or action.

The filter command parameters are described in Table 36.

Table 36 Filter Command Parameters

Parameter	Description
-i --ignore	Ignores case distinctions in both the pattern and the input.
-v --invert	Inverts the sense of matching, to select non-matching lines.



Parameter	Description
-A --after	Displays the number of lines of trailing context after matching lines.
-B --before	Displays the number of lines of leading context before matching lines.
<value>	The value for the number lines to be displayed after leading/trailing context.
<pattern>	A regular expression used for filtering the output of a command/action. It can have alphanumeric characters and special characters supported by POSIX regular expressions.

```
(config-SystemFunctions=1)>show | filter m
```

Example 154 Filter Output of Command show Matching a Pattern

```
(config-SystemFunctions=1)>show | filter -i s
```

Example 155 Filter Output with Case-Insensitive Match

```
(config-SystemFunctions=1)>show | filter -A 2 Fm
```

Example 156 Filter Output with Option -A

```
(config-SystemFunctions=1)>show | filter -A 3 Fil | filter -B 10 -I S
```

Example 157 Filter Output of Another Filter Command

[illegible]

Command **modify** `<modification_operations>` is a pipe extension command that can be used in Config mode to perform the same update operations on a set of selected MO instances. The command receives LDNs of MO instances as pipe input, and the operations as parameters.

To generate a set of LDNs of MO instances, command **show** combined with its search parameters is typically used:

```
show [-r|--recursive] [-m|--moc] <MO_class_name> [[-c|--condition]
<condition>] [-p|--property]
```

For more information about how to filter MO instances, see [Section 3.3.9 Filter MO Information](#) on page 63.

The command applies the following modifications to each MO instance that is piped to the command.

Syntax for attribute modification:

```

modify <attribute_name>=<attribute_value>
modify <attribute_name>['@<index>']='<attribute_value>
modify <attribute_name>=['<attribute_value>']
modify <attribute_name>['<old_attribute_value>']='<new_attribute_value>
modify <attribute_name>['@<index>']='<new attribute value>

```

Many attribute operations can be combined, see Example 159.

```
modify attribute1=xx attribute2=[yy,zz]
```

Example 159 Combination of Attribute Operations



Syntax for struct modification:

```
struct ::= <struct_attribute_name> | <struct_attribute_name>=<key_value> | <struct_attribute_name>['@<index>']
modify <struct>
modify <struct>,<member_name>=<member_value>
modify <struct>,<member_name>['@<index>']='<new_member_value>
modify <struct>,<member_name>=['<member_values>']
modify <struct>,<member_name>['<old_member_value>']='<new_member_value>
```

Many struct member operations can be combined, see Example 160.

```
modify aStruct=key,member1=xx member2[@1]=yy
modify aStruct=[@1],member1=xx member2[@1]=yy
```

Example 160 Combination of struct Member Operations

When executed, the operations performed on each MO instance are echoed.

For an example of activate and set a high priority on all PM jobs belonging to the group latency, see Example 161.

```
(config)>show -r -m PmJob -c jobGroup==Latency -p | modify request
edJobState=ACTIVE jobPriority=HIGH
ManagedElement=NODE06ST, SystemFunctions=1, Pm=1, PmJob=X, requestedJobState=ACTIVE jobPriority=HIGH
ManagedElement=NODE06ST, SystemFunctions=1, Pm=1, PmJob=Y, requestedJobState=ACTIVE jobPriority=HIGH
```

Example 161 Activate and Set High Priority on All PM Jobs Belonging to the Group Latency

3.13.3 Action Command

Command **action** <action_name> ('--'<param_name> <param_value>)* is a pipe extension command that can be used in Config mode to start the same action on a set of selected MO instances. The command receives LDNs of MO instances as pipe input, and the action as parameter.

To generate a set of LDNs of MO instances, command **show** combined with its search parameters is typically used, as described in Section 3.13.2 Modify Command on page 115.

The command starts the action on each MO instance that is piped to the command.

When executed, the resulting operation performed on each MO instance, and the result, are echoed.

For an example of delete all blob files from statistics directories in File Management, see Example 162.



```
(config)>show -r -m FileGroup -c fileGroupId=~".*statistics$" -p
| action removeFile --file blob
ManagedElement=NODE06ST,SystemFunctions=1,FileM=1,FileGroup=linkstatistics,removeFile --file blob
true
ManagedElement=NODE06ST,SystemFunctions=1,FileM=1,FileGroup=loadstatistics,removeFile --file blob
false
```

Example 162 Delete All Blob Files from Statistics Directories in File Management

3.13.4 No Command

Command **no** [**<modification operation>**] is a pipe extension command that can be used in Config mode to remove a set of selected MO instances, or values inside a set of selected MO instances. The command receives LDNs of MO instances as pipe input, and any contents as parameter.

To generate a set of LDNs of MO instances, command **show** combined with its search parameters is typically used, as described in Section 3.13.2 Modify Command on page 115.

The command applies a “no command” to each MO LDN that is piped to the command.

Syntax for deleting MO instances:

```
no
```

Syntax for deleting attributes values:

```
no <attribute_name>
no <attribute_name>=<attribute_value>
no <attribute_name>'['@<index>']'
```

Syntax for deleting struct member values:

```
struct ::= <struct_attribute_name> | <struct_attribute_name>=<key_value> | <struct_attribute_name>'['@<index>']'
no <struct>,<member_name>
no <struct>,<member_name>'['@<index>']'
no <struct>,<member_name>=<member_value>
```

An example of unsetting a struct member is shown in Example 163.

```
no aStruct[@1],aMember
```

Example 163 Unset a struct Member

An example of removing a value from a struct member is shown in Example 164.

```
no aStruct=aKey,aMember=aValue
```

Example 164 Remove a Value From a Struct Member



When executed, the resulting operations performed on each MO instance are echoed.

For examples of remove operations, see Example 165 and Example 166.

```
(config)>show -r -m SnmpTargetV1 -p | no
no ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1, SnmpTargetV1=1
no ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1, SnmpTargetV1=2
```

Example 165 Remove All SNMPv1 Targets

```
(config)>show -r -m FileGroupPolicy -c maxNumberFiles>0 | no maxFileSize
no ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, FileGroupPolicy=1, maxFileSize
no ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, FileGroupPolicy=10, maxFileSize
```

Example 166 Remove Maximum File Group Size Limitation in All File Group Policies Having a Maximum Number of Files Limitation

3.13.5 Format Command

The format command, (**<command> | <action> | format**), is a pipe extension command, which transforms the output of any command or action with CLI character escape sequences to actual characters in its output. The CLI character escape sequences supported are `\n`, `\r`, `\t`, `\"`, and `\\`.

The targeted strings are attribute values and complex action results.

A valid assumption for attribute values is that the concerned strings are always quoted. Since they are also either appeared after an attribute name and/or indented, this command adjusts the left margin of all lines to the position of the opening quote in its output as mentioned in Example 167.

If the user set the attribute value that contains formatting information, then CLI displays the output with actual characters.

```
(config-AMoc=1)>show userLabel
```

Example 167 Format Command

3.14 Help Command

Command **help** provides online help on both CLI modes and commands, and on the information model.



3.14.1 Static Help in CLI

Static help for the CLI can be retrieved by executing command **help** without any parameters. This command can be executed in both Exec and Config modes in the CLI.

The command provides information on the current mode and the various key bindings in the CLI. Along with the help provided on the CLI elements, the CLI state diagram is displayed when the command is executed, see Figure 1.

The output of command **help** has the CLI commands classified based on their purpose of use and operation. The commands are classified under categories GENERAL, NAVIGATION, CONFIGURATION, TRANSACTION, PIPE, and SPECIAL.

3.14.2 Information Model Help

Information model help is provided with the following commands:

- Command **help -t|--tree [-m|--moc <moc_name>]** displays the MOC tree.

Without a MOC parameter, the whole MOC tree from the top level MOC ManagedElement is displayed. With a MOC parameter, only the subtree beneath the specified MOC is displayed. If a MOC can occur in multiple locations in the tree, all subtrees are displayed. If a MOC can be nested beneath itself, only the top location is displayed.

- Command **help -m|--moc <moc_name> [-p|--property]** displays detailed information on a certain MOC, similar to how MOCs are documented in the online MOM descriptions.

The output displays the locations where a MOC can occur, its cardinality, and if it can be created and deleted by users. All possible child MOCs are listed, followed by the textual description of the MOC. Finally, all actions and attributes belonging to the MOC are listed. The list of actions and attributes can be suppressed by adding a property parameter without arguments.

- Command **help -m|--moc <moc_name> [-p|--property <property_name> [, <property_name>]...]** displays detailed information on one or more actions or attributes belonging to a MOC.

For actions, the output displays the textual description, the return type, and the parameters and their multiplicities and types. For attributes, the output displays their multiplicity, the textual description, and type information.

3.15 Deprecated Actions

The deprecated syntax for action execution is [**<path>**]**<action_name>**[**<action_parameter_value>**]**<action_parameter_value>** ...].



Auto-completion and help text for this syntax is not supported.

3.16 Deprecated Options

The options **all** and **verbose** are deprecated. They are replaced by the options **-r|--recursive** and **-v|--verbose**, respectively.

The option configuration is replaced by command **show-config**. The functionality of the new options is the same as the deprecated options.

Auto-completion and help request for the deprecated options is not supported, but command execution works:

- **show all** <Tab>
- **show config?**
- **show verbose**

3.17 CLI Commands Limitations

The CLI command functionality has the following limitations:

- Command **show-config** without option **-v|--verbose** cannot be used to copy MOs that contain mandatory keyless structs.
- If there are sequences of strings, or sequences of structs with string key attributes, and character **@** is displayed as the first character in these strings, the commands used to address these strings or structs must be performed using quoted strings.
- Help for **-c** and **-g** parameters in **show-counters** is displayed only if the complete PM group Id or measurement type/counter name is given as input.



4 Terminal Properties

This section describes the terminal properties.

4.1 Terminal Types

Some aspects of how the CLI interacts with the attached terminal or terminal emulator can be controlled by setting the terminal type. The CLI recognizes all terminal types supported by the operating system of the management system and also the following special terminal types:

- `dumb`

The CLI acts as if there is no terminal attached, but instead a script being fed to it on `stdin`. No prompts are printed and the input command lines are not echoed back. Only the command output is printed.

- `ossic-eam`

The CLI acts as if the attached terminal is a VT100, with the following exception:

A VT100 terminal, as well as terminals and terminal emulators compatible with it, do not show the cursor when an input line reaches the end of the terminal width. Thus, the CLI for most terminal types echoes an extra space followed by a backspace as an input line reaches this length. For terminal type `ossic-eam`, these extra characters are not echoed.

The terminal type is set by assigning a value to the `TERM` environment variable in the environment where `cliss` starts. The SSH protocol allows `TERM` to be set, but the methods depend on the client being used.

4.2 Default Key Bindings

Apart from the standard line edition keys such as **Left arrow**, **Right arrow**, **Backspace**, and **Delete**, the CLI supports several key combinations that can be used for editing the commands.

Support for key bindings is specific for the terminal type. The terminal settings override any key or key combination.

The CLI is aligned with the Libtecla library, which defines each key binding as follows:

- **Ctrl**

Press the **Ctrl** key and enter the next key simultaneously.



— Esc

Press the **Esc** key on the terminal followed by the next key, or press the **Meta** key simultaneously with the next key.

4.2.1 Move Cursor

The key combinations for moving the cursor are shown in Table 37.

Table 37 Move Cursor

Action	Key Combination
Move the cursor back one character	Ctrl+B or Left arrow
Move the cursor back one word	Esc+B or Alt+B
Move the cursor forward one character	Ctrl+F or Right arrow
Move the cursor forward one word	Esc+F or Alt+F
Move the cursor to the beginning of the command line	Ctrl+A
Move the cursor to the end of the command line	Ctrl+E

4.2.2 Delete Characters

The key combinations for deleting characters are shown in Table 38.

Table 38 Delete Characters

Action	Key Combination
Delete the character before the cursor	Ctrl+H
When the cursor is within the line, it deletes the cursor character. When began at the end of the line, it displays all possible completions then redisplay the line.	Ctrl+D
Delete all characters from the cursor to the end of the line	Ctrl+K
Delete the whole line	Ctrl+U
Delete the word before the cursor	Esc+Backspace or Alt+Backspace
Delete the characters between the last mark that was set and the cursor	Ctrl+W
Delete the word after the cursor	Esc+D or Alt+D

4.2.3 Insert Recently Deleted Text

The key combination for inserting recently deleted text is shown in Table 39.



Table 39 Insert Recently Deleted Text

Action	Key Combination
Insert the most recently deleted text at the cursor	Ctrl+Y

4.2.4 Display Previous Command Lines

The key combinations for displaying previous command lines are shown in Table 40.

Table 40 Display Previous Command Lines

Action	Key Combination
Scroll backward through the command history	Ctrl+P or Up arrow
Scroll forward through the command history	Ctrl+N or Down arrow

4.2.5 Capitalization

The key combinations for changing capitalization are shown in Table 41.

Table 41 Capitalization

Action	Key Combination
Capitalize the word at the cursor, that is, make the first character uppercase and the rest of the word lower case	Esc+C
Convert all characters of the word that follows the cursor to lower case	Esc+L
Convert all characters of the word that follows the cursor to upper case	Esc+U

4.2.6 Special Actions

More key combinations are shown in Table 42.

Table 42 Special Actions

Action	Key Combination
Abort a command or clear line	Ctrl+C
Arrange for the next character to be treated as a normal character This action allows control characters to be entered	Ctrl+V
Clear the terminal, then redisplay the current line	Ctrl+L



Table 42 Special Actions

Action	Key Combination
Swap the character under the cursor with the character just before the cursor	Ctrl+T
Redisplay the line	Ctrl+R