

Ericsson Command-Line Interface

INTERWORK DESCRIPTION

Copyright

© Ericsson AB 2014, 2015. 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
2	ECLI Concepts	5
2.1	ECLI Modes	5
2.2	ECLI Session	6
2.3	ECLI Transaction	7
2.4	ECLI Position	11
2.5	ECLI Prompt	14
2.6	ECLI Help	14
2.7	Auto-Completion	19
2.8	Case Correction	25
2.9	Escaping of Special Characters	26
2.10	Visibility Levels	27
2.11	Automatic Correction of ManagedElement ID	31
2.12	Limitations	31
3	ECLI Commands	33
3.1	Summary of ECLI Commands	33
3.2	Success and Error Indications	40
3.3	Display Information	43
3.4	Change and Display the Position in MO Tree	62
3.5	Display MO Instances	64
3.6	Change MO Attributes	67
3.7	Create MO	87
3.8	Reset MO	91
3.9	Delete MO	95
3.10	MO Actions	96
3.11	Copy and Paste Configuration Data	100
3.12	Change Password Command	100
3.13	Display ECLI Version	101
3.14	Pipe Utility Commands	102



3.15	Deprecated Actions	104
3.16	Deprecated Options	104
3.17	ECLI Commands Limitations	104
4	Terminal Properties	107
4.1	Terminal Types	107
4.2	Default Key Bindings	107



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 19.
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 25.
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 ECLI Commands on page 33
Configuration export and import	With command <code>show-config</code> , 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 3.5.4 Display Single MO and Its Child MOs in Configuration Printout Format on page 65.



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 ECLI Help on page 14.
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 ECLI Modes on page 5.
ECLI prompt	The ECLI prompt can be configured and provides information on the ECLI mode and ECLI position. For details, see Section 2.5 ECLI Prompt on page 14.
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 62.
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 ECLI Transaction on page 7.

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 standard 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 107.
- The user has IP access to the Operation and Maintenance (O&M) address of the ME.



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



2 ECLI Concepts

This section describes the ECLI concepts.

2.1 ECLI Modes

The ECLI provides the two following two 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 an ECLI session, the user always enter Exec mode, which is the default mode.

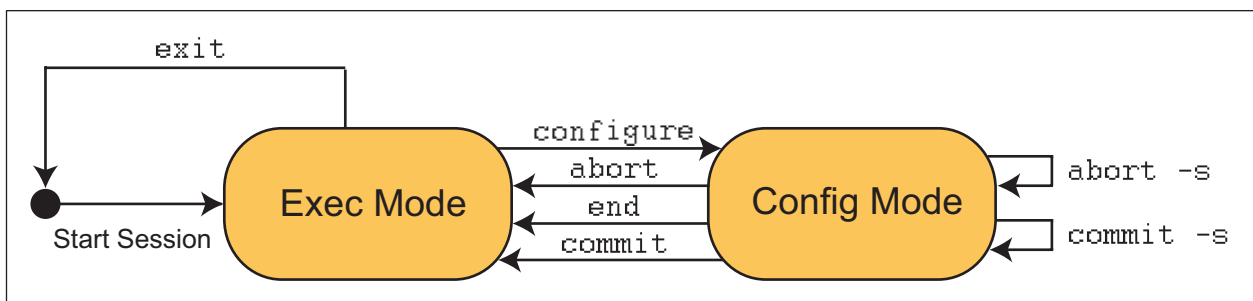


Figure 1 ECLI Modes

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



2.2 ECLI Session

This section describes an ECLI session.

2.2.1 Start an ECLI Session

To start an ECLI 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 MO.
- **-p** (port number) – TCP port 22 is default.

Root user access is denied.

2. Wait for the session to start.

2.2.2 Successful Logon

When logon is successful, the session starts in Exec mode, the ECLI startup message is displayed, and finally the Exec prompt > is displayed.

2.2.3 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 ECLI service is running.
subsystem request failed on channel 0	Check the SSH server state and verify that its configuration is valid.



2.2.4 Parallel Sessions

The maximal number of parallel sessions is 256.

2.2.5 Session Inactivity Timer

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

The default inactivity timer value is 120 seconds.

2.2.6 ECLI Session End

An ECLI session is closed either as a result of command `exit` or when the session inactivity timer expires.

2.3 ECLI 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 ECLI session starts in Exec mode and can be changed to Config mode by command `configure`.

If the ECLI 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 ECLI position is unchanged, unless the MO in the ECLI position no longer exists; in this case the ECLI position moves to the closest instantiated parent MO location.

Command `commit` without any argument performs a commit and the ECLI mode changes to Exec mode.

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

Example 1 Command commit

Command **commit -s** or **commit --stay** performs a commit, the ECLI 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  
(config-ManagedElement=NODE06ST) >
```

Example 2 Command commit -s

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

From the ECLI 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.

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

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 (<MO>) <error_specific_information_why_the_validation_failed>	The error-specific part gives detailed information about why the validation failed.
ERROR: Current cardinality of <current_cardinality> for class-instance of <MOC> is <<minimum_cardinality_value> (lower-limit) for <DN_of_validated_MO>	The error description explains why the command 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 <code>commit</code> 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 42. The transaction is still valid and its state is the same as before command <code>commit</code> 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="Stockholm, Building 25"
(config-ManagedElement=NODE06ST) >abort
(ManagedElement=NODE06ST) >
```

Example 3 Command abort

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

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

Example 4 Command abort -s

If the abort fails, this results in an error message and the ECLI 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 ECLI position is unchanged, unless the MO in the ECLI position no longer exists; in this case the ECLI 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
(ManagedElement=NODE06ST) >
```

Example 5 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 ECLI session. This command is valid only in the Exec mode.

2.4 ECLI Position

The ECLI position is a Distinguished Name (DN) of an MO instance identifying a position in the MO tree. The ECLI 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.

```
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1
```

Example 6 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.

```
ManagedElement=NODE06ST
ManagedElement=NODE06ST, SystemFunctions=1,
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1
ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1
```

Example 7 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 ECLI Path

An ECLI 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 <i>ManagedElement</i> . Indicated in the DN and <i><path></i> field of the ECLI prompt as an empty string.
Position at ManagedElement=N ODE06ST	The position at <i>ManagedElement</i> . Indicated in the DN and RDN field of the ECLI prompt as ManagedElement=NODE06ST.
Position at SystemFunctions =1	The position at <i>SystemFunctions</i> . Indicated in the RDN field of the ECLI prompt as SystemFunctions=1. Indicated in the DN field of the ECLI prompt as ManagedElement=NODE06ST, SystemFunctions=1.
Position at SysM=1	The position at <i>SysM</i> . Indicated in the RDN field of the ECLI prompt as SysM=1. Indicated in the DN field of the ECLI prompt as ManagedElement=NODE06ST, SystemFunctions=1, SysM=1.

2.4.6 Interpretation of ECLI Commands

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

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

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

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

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

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



2.5 ECLI Prompt

The ECLI prompt provides information on the status and context of the ECLI 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 ECLI modes for the lifetime of the session to contain any elements shown in Table 7.

Table 7 ECLI Prompt Configuration Elements

Element	Description
<code>\$default</code>	The <code>\$default</code> value is equivalent to <code>\$mode-\$rdn</code> in Config mode and an empty string in Exec mode.
<code>\$dn</code>	LDN indicating the ECLI position relative to the root position.
<code>\$mode</code>	ECLI mode, that is, <code>config</code> or <code>exec</code> .
<code>\$hostname</code>	The name of the host the ECLI service runs on.
<code>\$nodename</code>	The key value of the <i>ManagedElement</i> MO.
<code>\$rdn</code>	RDN indicating the ECLI position relative to the parent MO.
<code>\$user</code>	User account name.
<code><Value: Any user-defined string></code>	<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 <code>\"</code>, <code>\'</code>, <code>\t</code>, <code>\n</code>, <code>\b</code>, <code>\f</code>, <code>\r</code>, and <code>\v</code>.</p> <p>Special characters with escaped hexadecimal form are also supported, in form <code>\x</code>, followed by the ASCII code in hexadecimal in two digits. For example, the hash character <code>#</code> must be specified as <code>\x23</code> and is displayed in the ECLI prompt as <code>#</code>.</p>

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

2.6 ECLI Help

The ECLI 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 ECLI operations and MOM elements (MOCs, attributes, and actions) that are available for the following:

- ECLI mode
- ECLI position
- ECLI 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 ECLI 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 ECLI 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 (ECLI 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).

ECLI help is not available for the position in key values of the ECLI path. For example, no ECLI 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.

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

ECLI Element	Printout Syntax
Action	<i><action_name></i> () <i><spaces></i> <i><truncated_action_description></i>
Action parameter	<i><parameter_name></i> <i><spaces></i> <i><truncated_parameter_description></i>
Class	+ <i><class_name></i> <i><spaces></i> <i><truncated_class_description></i>



Table 8 Printout Syntax for Brief Help

ECLI Element	Printout Syntax
ECLI 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_descripti on></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 8 and Example 9.

```
(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 8 Brief Help

```
(config-MOex1=1)>addNumbers ?
--num1                Number one
--num2                Number two
```

Example 9 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)>?
actionA() first line of help text
myMultiValueAttr[] first line of help text
+B first line of help text
```

Example 10 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	<code><action_name>() [Return Type]</code> <code><action_description></code>
Action parameter	<code><parameter_name><parameter_type> [passphrase]</code> <code>[optional/default=<default_value>]</code> <code><parameter_description></code>
Class	<code><class_name> MO type⁽¹⁾ [singleton] [optional]</code> <code><class_description></code>
ECLI operation	<code><ECLI_operation_name>() Command</code> <code><operation_description></code>
ECLI operation parameter	<code><parameter_name></code> <code><parameter_description></code>
Single-valued attribute, struct member	<code><attribute_name> <attribute_type> [passphrase] [optional/read only/default=<default_value>/exclusive]</code> <code><attribute_description>⁽²⁾ <attribute_description>⁽²⁾</code>

(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 44.

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 11 Verbose Help on MOC

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

Example 12 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 13 Verbose Help on String Attribute

If help is triggered directly after (without space) the ECLI operation or action name, Verbose Help is provided on the operation, as shown in Example 14.

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

Example 14 Verbose Help on ECLI 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 15 through Example 19.

```
(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 15 Verbose Help on ECLI Command

```
(config-MOex2=1)>concatString_defValues --str1 ?
--str1      String [optional/default=com]
String one
```

Example 16 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 17 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 18 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 19 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 20 Verbose Help on Filter

2.6.3 Static Help

Command **help** provides static help about the ECLI. This command can be executed in both Exec mode and Config mode.

The help information consists of an ECLI state diagram and summary of the main commands. It also gives information on the key bindings in the ECLI.

In the help information, the ECLI commands are classified based on their purpose of use and operation. The commands are classified under categories **GENERAL**, **NAVIGATION**, **CONFIGURATION**, and **TRANSACTION**.

2.7 Auto-Completion

By pressing the **Tab** key at any position of the ECLI command, completion can be requested on all possible continuations of the ECLI 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 ECLI command are displayed in Completion Possibility List with the following content and in the following order:</p> <ul style="list-style-type: none">• ECLI command name, for example, show or history.• ECLI 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 ECLI 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	<p>Auto-completion automatically adds them, including the following elements:</p> <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 ECLI command is invalid	Auto-completion or a completion possibility list is not provided.

Auto-completion is provided only for valid ECLI 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
Comma (,)	Completes DNs if they can be followed by a further MOC name or DN.
Enter (CR)	Completes operation, attribute, action names, and DNs and then executes the completed command, if there is a unique match.
Equal sign (=)	Completes DN and attribute names if they can be followed by an = character.
Space (" ")	Completes command names that can be followed by parameters.
Tab	<p>When pressed at any position of the ECLI command, all possible continuations of the command and MOM-defined elements the user is authorized to are displayed. Provides auto-completion or a completion possibility list. Example:</p> <pre>(Pm=1) ><Tab> PmGroup PmMeasurementCapabilities=1 configure dn exit help history length prompt scriptmode show show-config show-dn show-mib show-table top up width</pre>

2.7.2 Examples of Completion Possibility Lists

This section provides examples of completion possibility lists.



```
(config)>show<Tab>
show-config
show-dn
show-mib
show-table
<cr>
<space>
```

Example 21 Completion Possibility List of Command show without Space

```
(config-MOex3=1)>add<Tab>
addNumbers
addNumbers_defValues
addNumbersNameClash
addsoc
```

Example 22 Completion Possibility List for Actions without Space

```
(config-MOex4=1)>addNumbers --n<Tab>
--num1
--num2
```

Example 23 Completion Possibility List for Action Parameters

```
(config)>show <Tab>
--moc
--recursive
--sort
--verbose
-m
-r
-s
-v
ManagedElement=NODE06ST
<cr>
```

Example 24 Completion Possibility List of Command show with Space

```
(config-ManagedElement=NODE06ST)>show | filter <Tab>
--after
--before
--ignore
--invert
-A
-B
-I
-v
<pattern>
```

Example 25 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	<code>show M<Tab></code>	<code>show ManagedElement=NODE06ST</code>
	<code>show m,</code>	<code>show ManagedElement=NODE06ST,</code>
	<code>show m<Enter></code>	Auto-completion and execution of <code>show ManagedElement=NODE06ST</code>
	In Config mode in root position: Press <Enter> with empty command line	Navigation to ManagedElement=N ODE06ST
Unique match for partial DN	<code>show ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=A<Tab></code>	<code>show ManagedElement=NODE06ST, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=A</code> if the FileGroup=AlarmLogs and FileGroup=AlertLogs MOs exist
Unique match for operation name	In Exec mode: <code>c<Enter></code>	Triggers Config operations
Unique match for MOC	<code>show m=</code>	<code>show ManagedElement=</code>
Unique match for attribute	<code>ManagedElement=NODE06ST, u=</code>	<code>ManagedElement=NODE06ST, userLabel=</code>
Unique match for operation name	<code>show-c<Space></code>	<code>show-config</code>
Multiple matches for parameter names of an action	<code>addNumbers <Tab></code>	<code>addNumbers</code> <code>--num1</code> <code>--num2</code>
Unique match for value of a parameter of an action	<code>addNumbers --num1<Tab></code>	<code>addNumbers --num1</code> <code><value></code>
Unique match for parameter name of an action	<code>addNumbers --num1 20<Tab></code>	<code>addNumbers --num1 20</code> <code>--num2</code>
Multiple matches for <code><space></code> , <code><cr></code> , and <code><value></code> of a parameter with value of an action	<code>addNumbers defValues --num1 23<Tab></code>	<code>addNumbers_defValues</code> <code>--num1 23</code> <code><value></code> <code><space></code> <code><cr></code>



2.7.3.1 Auto-Completion of MO Reference Type

The input must be quoted to trigger completion for MO references. For example, assume a model with MO D under MO C, MO C under MO B, MO B under the root ManagedElement=NODE06ST, and MoReferenceAttribute under MO C. The related output is shown in Example 26.

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

Example 26 Auto-Completion of MO Reference Type

A relative path is always relative to the ECLI position. For example, assume a model with MO C1 and MO C2 under MO B, MO B under the root ManagedElement=NODE06ST, and MoReferenceAttribute under MO C1. The related output is shown in Example 27.

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

Example 27 Auto-Completion of MO Reference Type

2.8 Case Correction

A unique match of auto-completion automatically triggers case correction (for example, LogicalFS to LogicalFs) of one ECLI 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
- ECLI operation and parameter name
- Action parameter name

Case correction for string attribute values is not supported.



2.9 Escaping of Special Characters

The ECLI 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 ECLI, 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
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\"</code>	Quotation mark
<code>\\</code>	Escape character
<code>\?</code>	Question mark ⁽¹⁾
<code>\!</code>	Exclamation mark ⁽¹⁾
<code>\#</code>	Hash ⁽¹⁾
<code>\xNN</code>	Any character in hexadecimal format (2 hexadecimal digits). N is any character 0–9, A–F, or a–f. The sequence <code>\x00</code> , which translates to the string termination character in C, is not allowed. Non-US-ASCII characters are not allowed (<code>NN</code> is greater than <code>\x7F</code>).
<code>\NN</code>	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=1)>userLabel=hello\nworld
(config-ManagedElement=1)>show userLabel
userLabel="hello\nworld"
```

Example 28 *Set String Attribute with Special Character*



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

Example 29 Set Struct Member with Special Character

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

```
(config-ManagedElement=1)>userLabel=temp\76value
ERROR: Invalid value 'temp\76value' for attribute
'userLabel'.
This is not a valid escape sequence
```

Example 30 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 ECLI, 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 Visibility Levels

The behavior of ECLI 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 ECLI 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 <code>Element not visible</code> 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 <code>Element not visible</code> 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



Table 14 Visibility Behavior

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

2.10.1 Auto-Completion

Auto-completion for accessible element `YDeprecatedThing` is not supported, as the current location is visible element `YCurrentThing`, as shown in Example 31.

```
(config-YCurrentThing=1) >YDepre<Tab>
```

Example 31 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 32.

```
(config-YDeprecatedThing=1) >YSample<Tab>
YSampleDeprecatedThing=1
```

Example 32 Auto-Completion of Accessible Element

2.10.2 Display Information

When command `show` is executed from the visible current location, only visible elements, such as `YCurrentThing`, are displayed. as shown in Example 33.

```
(config-YCurrentThing=1) >show
YCurrentThing=1
  userLabel="UserLabel"
```

Example 33 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 34.

```
(config-YCurrentThing=1)>show -v
YCurrentThing=1
  rwattr1=[] <empty>
  userLabel="UserLabelValue"
  yCurrentThingId="1"
  YDeprecatedThing=1 <deprecated>
```

Example 34 Command show -v

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 35.

```
(config-YThing=1)>show-config
YThing=1
  restrictedattr=4
  rwattr1="RW-One"
  rwattr2=2
  rwattr3=3
  YCurrentThing=1
    userLabel="UserLabelValue"
    YDeprecatedThing=1
      depAttr1=UNLOCKED
      depAttr2=16
      deprecatedStruct
        bool1=true
        int1=132
        str1="StringValue1"
        str2="StringValue2"
      up
      depStructWithKeyAttribute="StringValue1"
        bool1=true
        int1=132
        str2="StringValue2"
      up
    YSampleDeprecatedThing=1
      rwattr2=2
    up
  up
up
```

Example 35 Command show-config



2.10.3 Help

For accessible elements, help information is displayed when the user specifies a full element name, but not when only a partial element name is specified.

When the life cycle status of an element is anything else than the current one, the status is included as a tag in the help information, as shown in Example 36.

```
(config-YCurrentThing=1)>YDeprecatedThing?  
YDeprecatedThing MO Type [optional] [deprecated]  
aMOC
```

Example 36 Help for Accessible Element

For not-visible elements, no help is supported.

2.11 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 ECLI scripts and other tools to be used for several nodes, the ECLI accepts any 3GPP® compliant *ManagedElement* ID as part of a DN. The ECLI can automatically correct this to the actual/correct ID.

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

2.12 Limitations

The ECLI has the following limitations:

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





3 ECLI Commands

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

Table 15 ECLI Commands

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

3.1 Summary of ECLI Commands

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

Table 16 Summary of ECLI Commands

Command	Mode	Description
<code>?</code>	Exec Config	Displays context-sensitive help on ECLI operations and MOM elements. For details, see Section 2.6 ECLI Help on page 14.
<code>abort [-s --stay]</code>	Config	Discards changes in the transaction and terminates the transaction. For details, see Section 2.3.3 Abort on page 10.



Table 16 Summary of ECLI Commands

Command	Mode	Description
<code>back [-h --history]</code>	Exec Config	Navigates back to the previous position in the MO tree with the following option: <ul style="list-style-type: none">• <code>-h --history</code> – 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 <code>top</code> can be used instead.
<code>commit [-s --stay]</code>	Config	Validates the transaction and, on success, commits the configuration changes. For details, see Section 2.3.2 Commit on page 7.
<code>configure</code>	Exec	Changes the ECLI mode from Exec mode to Config mode.
<code>dn <LDN></code> <code>dn -m --moc <moc_name></code> <code>[-c --condition <condition></code>	Exec Config	Navigates to any location in the MOM with the following options: <ul style="list-style-type: none">• <code>-c --condition</code> – Criteria for the MO to navigate to. This can be used if there is more than one instance of the specified MOC.• <code>-m --moc</code> – The MOC to navigate to. For details, see Section 3.3.9.1 Parameters to Filter MO Information on page 54.
<code>end</code>	Config	Changes the ECLI mode to Exec mode, if there are no changes in the transaction. For details, see Section 2.3.4 End on page 10.
<code>exit</code>	Exec	Exits the ECLI session. For details, see Section 2.2.6 ECLI Session End on page 7.



Table 16 Summary of ECLI Commands

<code><command> <action> filter [-i --ignore] [-v --invert] [{-A --after} <value>] [{-B --before} <value>] <pattern></code>	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 3.14.1 Filter Command on page 102.</p>
help	Exec Config	<p>Provides static help on current ECLI mode and available commands in this mode. For details, see Section 2.6.3 Static Help on page 19.</p>
<code>history [-s --size] [<number>]</code>	Exec Config	<p>Without parameter, displays the command history of the ECLI session in chronological order in format <code><sequence_number> <date> <time> <command></code> 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 ECLI Commands

<code>insert [<path>,<attribute_name>['<existing_sequence_element>'⁽¹⁾]=<new_sequence_element></code>	Config	Inserts simple type elements to a sequence or inserts an struct element to a sequence of keyless structs. For details, see Section 3.6.3.3 Insert Simple Type Elements to Sequence on page 71, Section 3.6.3.4 Insert Simple Type Element in Sequence Using Index on page 72, and Section 3.6.3.13 Insert Struct Element to a Sequence of Keyless Structs on page 80.
<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 ECLI output rows until <code>--More--</code> 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 3.9 Delete MO on page 95, Section 3.6.2 Delete Value of Single-Valued Attribute on page 70, Section 3.6.3.7 Delete Named Element from Sequence on page 75, Section 3.6.3.8 Delete Element in Sequence Using Index on page 75, Section 3.6.3.9 Delete All Elements from Sequence on page 76, and Section 3.6.3.12 Delete Keyless Struct Sequence on page 79.
<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 ECLI user password in root position. It is available only if the ME supports changing user password through the ECLI. For details, see Section 3.12 Change Password Command on page 100.
<code>[<path>,<action_name>[<action_name>[<action_parameter_name>[<action_parameter_value>]] ...]</code>	Exec Config	Requests action execution. For details, see Section 3.10.1 Action Request on page 96.
<code>[<path>,<attribute_name>=<attribute_value></code>	Config	Assigns value to an attribute. For details, see Section 3.6.1 Change Single-Valued Attribute on page 68.



Table 16 Summary of ECLI Commands

<code>[<path>,] <attribute_name> ['<existing_sequence_element>'] '='<new_sequence_element_value></code> (1)	Config	Changes or adds a sequence element. For details, see Section 3.6.3.5 Change Sequence Element on page 73 and Section 3.6.3.6 Change or Add Sequence Element Using Index on page 73.
<code>[<path>,] <attribute_name> ['@<index>'] '='<new_value></code> (2)		
<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 ECLI session. For details, see Table 7.
<code><RDN></code>	Exec Config	Changes the ECLI position to the RDN. If the MO does not exist, the MO is created in Config mode. For details, see Section 3.7 Create MO on page 87.
<code>reset [. <attribute_name>]</code>	Config	Resets an MO, an attribute, or a struct member to its original state. For details, see Section 3.8 Reset MO on page 91.
<code>scriptmode [--on --off]</code>	Exec Config	<p>Without parameter, displays the present state of the <code>scriptmode</code> in the ongoing ECLI session.</p> <p>Parameter <code>--on</code> turns on <code>scriptmode</code>, in which help function, auto-completion, case correction, and page break is disabled in ongoing ECLI session if not done.</p> <p>Parameter <code>--off</code> turns off <code>scriptmode</code>, in which help function, auto-completion, case correction, and page break is enabled back in ongoing ECLI session if not done already.</p>

Table 16 Summary of ECLI 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.
<pre>show [-s --sort] [<path>] -m --moc <moc_name> [-p --property <attribute_name>] [,<attribute_name>] ...] [-r --recursive] [-v --verbose] [-c --condition <condition>]</pre>		<ul style="list-style-type: none"> • -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 54. • -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 12 and Section 2.4.3 Relative Distinguished Name on page 12. • Without options, only those attributes are displayed having a value assigned and not a default value. <p>For details, see Section 3.3 Display Information on page 43.</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 45.



Table 16 Summary of ECLI Commands

show-counters [<i><DN></i>] [-j --pmJob <i><job_id></i>] [-v --verbose] [-c --counters <i><counter></i> [, <i><counter></i>] ...]	Exec Config	<p>Displays real-time values of Performance Management (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 <i>PmJob</i>. • -v --verbose – Displays verbose information. • <i><DN></i> – Selects the MO instance to show counters from. If omitted, counters from the current MO are displayed. <p>This command is available only if the ME supports displaying measurements through the ECLI. For details, see Section 3.3.9.5 Display PM Measurements on page 60.</p>
show-dn	Exec Config	Displays the user location in the MOM.
show-mib [-v --verbose] [-s --sort] [<i><path></i>]	Exec Config	Displays MO instance information. For details, see Section 3.3.8 Display MO Instance Information on page 54.
show-table [-r --recursive] [<i><path></i>] -m --moc <i><moc_name></i> [-p --property <i><attribute_name></i> [: <i><column_width></i>] [, <i><attribute_name></i> [: <i><column_width></i>]]] ...] [-c --condition <i><condition></i>] [-s --sort]	Exec Config	<p>Displays MO information in table format with the following options:</p> <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 54. • -p --property – Displays hidden attributes, include the hidden attributes. • -r --recursive – Implies that the displayed instances can be anywhere below the current position. <p>For details, see Section 3.3.9.3 Display MO Information in Tabular Format on page 58.</p>
top	Exec Config	Changes the ECLI position to the root position.
up	Exec Config	Changes the ECLI position to the parent MO.



Table 16 Summary of ECLI Commands

validate	Config	Validates the configuration changes in a transaction. Returns Transaction validation failed! or Transaction is valid!.
version	Exec Config	Displays the ECLI version. For details, see Section 3.13 Display ECLI Version on page 101.
width [<Width>]	Exec Config	<p>Without parameter, displays the number of ECLI output characters printed on a line until the line is broken. Primarily intended for informing the ECLI about the actual terminal width when the ECLI cannot determine it by itself. Setting the width to a value other than the actual terminal width is not recommended.</p> <p>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.</p>

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

```
ERROR: <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.
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



Table 17 Examples of Generic Error Messages

Error Message	Error Semantics
ERROR: Invalid value <i><issued command with arguments></i> for integer parameter <i><issued command with arguments></i> . 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 <i><MOC_name></i> are not creatable	The user tried to create an MO instance that cannot be created.
ERROR: Instances of <i><MOC_name></i> 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: <i><MO class path></i> <i><MO class path></i> A class path is required	There is more than one MOC with the same name when using parameter <i>-m</i> <i>--moc</i> . The MOC path is a comma-separated list of the class names, for example, ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1. This MOC path can be used as argument to parameter <i>-m</i> <i>--moc</i> .

3.2.1 Error Codes

The error codes shown in ECLI 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.



Table 18 Error Codes

Error Code	Description
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 <code>commit</code> .

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 44
- Configurational information, see Section 3.3.2 Display Configurational Information on page 45
- Struct, see Section 3.3.3 Display Struct on page 48
- Struct member, see Section 3.3.4 Display Struct Member on page 49
- Sequence of simple type, see Section 3.3.5 Display Sequence of Simple Type on page 50
- Sequence of struct, see Section 3.3.6 Display Sequence of Struct on page 51
- Sequence element of sequence of struct, see Section 3.3.7 Display Sequence Element of Sequence of Struct on page 52
- MO instance information, see Section 3.3.8 Display MO Instance Information on page 54
- MO information in tabular format, see Section 3.3.9.3 Display MO Information in Tabular Format on page 58
- PM measurements, see Section 3.3.9.5 Display PM Measurements on page 60



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

(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
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	MO reference is displayed as a string containing the LDN of the referred MO. Example: "ManagedElement=NODE06ST, SystemFunctions=1, MyMo=42"



Table 20 Display Formats of Supported Attribute Value Data Types

Attribute Value Data Type	Description
passphrase string	Displayed as a masked value (*****) or in encrypted form, depending on how the ME is configured. For details, see Section 3.6.5 Change Attribute Defined as Password or Passphrase String on page 83.
password	Displayed as a string in encrypted form. Password change is supported in a special way, see Section 3.6.5 Change Attribute Defined as Password or Passphrase String on page 83.
string	Displayed in double quotation marks. Example: "ABC"

```
>show ManagedElement=NODE06ST,userLabel
userLabel="BTS#21 in Zone C"
```

Example 37 Display Single-Valued Attribute

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

Example 38 Display Single-Valued Attribute for EcimEmpty

```
>show -r ManagementElement=NODE06ST,MOCex1=1,MOex
5=1,anAttrWithIntDerivedType
anAttrWithIntDerivedType=16
```

Example 39 Display Single-Valued Attribute by Command show -r

```
>show ManagementElement=NODE06ST,MOCex2=1,userLabel
userLabel="NODE"
```

Example 40 Display Single-Valued Attribute by Using LDN

3.3.2 Display Configurational Information

Command **show-config** [-v|--verbose] [-s|--sort] [<path>] displays the output in configuration format. It also automatically enables parameter **-r|--recursive**.

Command **show-config** with the verbose option displays attributes that are not set and attributes set to their default values.

The difference between `show-config` and `show-config` with the verbose option when copying the output into an ECLI, is as follows:

- With parameter `-v | --verbose`, the command overwrites, that is, deletes non-set (in the input) optional values and sets non-set default values back to default.

It is recommended to use `show-config` with `-v | --verbose` when copying the output into an ECLI.

- Without parameter `-v | --verbose`, the command appends, that is, only includes values that are set.

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

```
(config-MOex6=1) >show-config
MOex6=1
administrativeState=UNLOCKED
aManagedObject="ManagedElement=NODE06ST,MOCex3=1,MOex1=1"
anAttrWithIntDerivedType=16
anAttrWithStringDerivedType="xx6BBBBBBBBBBBxx"
aSimpleStruct
  bool1=true
  int1=132
  str1="StringValue1"
  str2="StringValue2"
  up
aStructWithDefValues
  up
aStructWithKeyAttribute="StringValue1"
  bool1=true
  int1=132
  str2="StringValue2"
  up
up
```

Example 41 Display Configuration Output without Verbose Information



```
(config-MOex7=1) >show-config -v
MOex7=1
administrativeState=UNLOCKED
aManagedObject="NODE06ST,MOCex4=1,MOex=1"
anAttrWithIntDerivedType=16
anAttrWithStringDerivedType="xx6BBBBBBBBBBBxx"
  no anAttrWithStringDerivedType_noDefault
  no anEcimEmpty
  no anyManagedObject
  no DateTimeTestValue
  no DateTimeWithoutOffsetTestValue
  defaultValue=0
  no DifferenceFromUTCTestValue
  no ipDNSAddressTestValue
  no noDontAutocompleteOnNo
  no ProblemCauseTestValue
  no rebelObject
  no RuleDataTypeTestValue
  no TimeoutTestValue
  no UnsignedRangeDataTestValue
  no anExclusiveStruct
  no anOptionalStruct
  no aPassphraseStruct
aSimpleStruct
  bool1=true
  int1=132
  int2=42
  str1="StringValue1"
  str2="StringValue2"
  up
aStructWithDefValues
  bool1=false
  int1=1
  str1="HoggaBogga"
  up
  no aStructWithHiddenAttribute
aStructWithKeyAttribute="StringValue1"
  bool1=true
  no int0
  int1=132
  int2=42
  str1="StringValue1"
  str2="StringValue2"
  up
no aStructWithMoRefs
no ecimStructArray
no MultiValueStructWithKeyAttr
up
```

Example 42 *Display Configuration Output with Verbose Information*

```
(Snmp=1) >show-config
Snmp=1
agentAddress
  host="0.0.0.0"
  port=26343
  up
agentAddress
  host="1.1.1.1"
  port=9999
  up
```

Example 43 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.

```
(Snmp=1) >show-config -v
Snmp=1
administrativeState=UNLOCKED
no nodeCredential
no trustCategory
agentAddress[@1]
  host="0.0.0.0"
  port=26343
  up
agentAddress[@2]
  host="1.1.1.1"
  port=9999
  up
```

Example 44 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-MOex8=1) >show aSimpleStruct
aSimpleStruct
  bool1=true
  int1=132
  str1="StringValue1"
  str2="StringValue2"
```

Example 45 Display Struct

```
(config-MOex9=1) >show --verbose aSimpleStruct
aSimpleStruct
  bool1=true
  int1=132
  int2=42 <default>
  str1="StringValue1"
  str2="StringValue2"
```

Example 46 Display Struct with ECLI Operation Parameter

```
(config-MOex10=1) >show -r aSimpleStruct
aSimpleStruct
  bool1=true
  int1=132
  str1="StringValue1"
  str2="StringValue2"
```

Example 47 Display Struct with ECLI 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-MOex11=1) >show aSimpleStruct,str1
str1="string"
```

Example 48 Display Struct Member

```
(config-MOex11=1) >show -v aSimpleStruct,int2
int2=42 <default>
```

Example 49 Display Struct Member with ECLI 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
intMultivalueMember
  42
  43
  44
```

Example 50 Display Sequence Structure Member

```
(config-structWithMultivalueMembers)>show -v
intMultivalueMember
intMultivalueMember <default>
  42
  43
  44
```

Example 51 Display Sequence Structure Member with ECLI 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]
intMultivalueMember
  43
```

Example 52 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
stringMultivalueAttr
  "STRING"
  "SWEDEN"
```

Example 53 Display Sequence of Strings

```
(config-MultivalueThing=1)>show -r PlainMultivalueAttr
stringMultivalueAttr
  "STRING"
  "SWEDEN"
```

Example 54 Display Sequence of Strings with ECLI Operation Parameter

```
(config-PlainMultivalueAttrs=1)>show -v intMultivalueAttr
intMultivalueAttr <default>
  42
  43
  44
```

Example 55 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
intMultivalueAttr
  42
  43
  44
(config-PlainMultivalueAttr=1)>show intMultivalueAttr[@1]
intMultivalueAttr
  42
```

Example 56 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
agentAddress
  host="1.1.1.1"
  port=111
agentAddress
  host="2.2.2.2"
  port=222

```

Example 57 Display Sequence of Structs without Key

```

>show -v ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1, agentAddress
agentAddress [@1]
  host="1.1.1.1"
  port=111
agentAddress [@2]
  host="2.2.2.2"
  port=222

```

Example 58 Display Sequence of Structs without Key with ECLI 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 59, 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
multiValueStructWithIntId=2
  capital="kol"
  id=2 <key>
  name="countryName" <default>
  population=0 <default>
multiValueStructWithIntId=3
  capital="xyz"
  id=3 <key>
  name="countryName" <default>
  population=0 <default>
multiValueStructWithIntId=1
  capital="delhi"
  id=1 <key>
  name="countryName" <default>
  population=0 <default>
(config-MultivalueThing=1)>show -v PlainMultivalueAttrs=1,=>
multiValueStructWithIntId=1
  capital="delhi"
  id=1 <key>
  name="countryName" <default>
  population=0 <default>
```

Example 59 *Display Sequence Element of Sequence of Structs with Key*

```
>show ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, S
nmp=1, agentAddress, host
ERROR: Multi-value struct attribute 'agentAddress' is missing
a key value or an index
```

Example 60 *Display Sequence Element of Sequence of Structs without Key*

```
(config-PlainMultivalueAttrs=1)>show -r multiValueStructWithIntId
multiValueStructWithIntId=1
  capital="Sweden"
  population=20000000
multiValueStructWithIntId=2
  capital="Norway"
```

Example 61 *Display Sequence Element of Sequence of Structs with Key with ECLI Operation Parameter*



3.3.8 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
SecM=1
  UserManagement=1
    LocalAuthorizationMethod=1
      CustomRole=CustomSystemOperator
      CustomRule=Custom_FaultManagement_1
      Role=NodeApplicationAdministrator
      Rule=NodeApplicationAdministrator
```

Example 62 Display MO Instance Information Recursively

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

```
(config-ManagedElement=NODE06ST) >show-mib -v MOCex5=1
ManagedElement=NODE06ST,MOCex5=1
ManagedElement=NODE06ST,MOCex5=1,MOex1=1
ManagedElement=NODE06ST,MOCex5=1,MOex2=1
ManagedElement=NODE06ST,MOCex5=1,MOex3=1
ManagedElement=NODE06ST,MOCex5=1,MOex4=1
ManagedElement=NODE06ST,MOCex5=1,MOex5=1
ManagedElement=NODE06ST,MOCex5=1,MOex6=1,MOex7=1
ManagedElement=NODE06ST,MOCex5=1,MOex8=1
```

Example 63 Display MO Instance Information with ECLI 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`. Up to two conditions can be specified for this option. The attribute types that can be used as part of the condition are integer, string, boolean, and enum.

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

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))
```



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

Example 64 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=1, SystemFunctions=1, SysM=1, Schema=ECIM_CommonLibrary
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComTop
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComSecM
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComLocalAuthorization
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComLdapAuthentication
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComSysM
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComFm
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComSnmp
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComFileM
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=CmwPm
```

Example 65 Display only Found Instances, no Attributes

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

Example 66 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=1, SystemFunctions=1, SysM=1, Schema=ComLdapAuthentication
  baseModelIdentifier="ECIM_LDAP_Authentication"
  baseModelVersion="2.0"
  identifier="ComLdapAuthentication"
  version="11.0.0"
ManagedElement=1, SystemFunctions=1, SysM=1, Schema=ComFileM
  baseModelIdentifier="ECIM_FileM"
  baseModelVersion="3.1.0"
  identifier="ComFileM"
  version="11.0.0"
```

Example 67 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 68 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 69 Specify Attributes and Multiple Conditions for Filtering MO Information

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 70 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 71 and Example 72.

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

Example 71 Display Single Attribute with Command show-table

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

Example 72 Display Multiple Attributes with Command show-table

If no value is available for an attribute, it displays as empty, as shown in Example 73.

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

Example 73 Display Empty Attribute with Command show-table

Display Attributes in Tabular Format Based on a Condition

Example 74 shows how to display attributes based on a condition.

```
(config-MOCex6=1)>show-table -m MOex9 -p administrativeState
,defaultValue -c (defaultValue==0)
=====
| administrativeState | defaultValue |
=====
| UNLOCKED           | 0           |
=====
```

Example 74 Display Attributes Based on Condition with Command show-table

Show-Table Width

Command `show-table` automatically sets its width based on the terminal width and the column width specified by the user. If the column width is not specified, it is set based on the largest item in the column.

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 <Tab>
administrativeState
agentAddress
operationalState
snmpId
(config-SysM=1)>show -m Snmp -p agentAddress, <Tab>
administrativeState
operationalState
snmpId
```

Example 75 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 <Tab>
administrativeState
operationalState
snmpId
(config-SysM=1)>show-table -m Snmp -p administrativeState, <Tab>
operationalState
snmpId
```

Example 76 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 available only if the ME supports displaying measurements through the ECLI.

Command `show-counters [<DN>] [-j | --pmJob <job_id>] [-v | --verbose] [-c | --counters <counter>[, <counter>] ...]` displays active Performance Management (PM) counters for an MO instance. The command works analogous to the other `show` commands, but displays PM counters associated with MO instances instead of attributes.



```
(MeasObj=1) >show-counters
MeasObj=1
  intCounterActive=123
  multivalueIntCounter
    12
    14 <suspect>
    16
  floatCounterActive=123.456 <suspect>
  multivalueFloatCounter
    1.345
    2.456
    3.5678 <suspect>
```

Example 77 Display PM Measurements with Command show-counters

```
(MeasObj=1) >show-counters -v
MeasObj=1
  intCounterActive=123 <PmJob=1> <Gp=15 min>
  multivalueIntCounter <PmJob=1> <Gp=15 min>
    12
    14 <suspect>
    16
  floatCounterActive=123.456 <PmJob=1> <Gp=15 min> <suspect>
  multivalueFloatCounter <PmJob=1> <Gp=15 min>
    1.345
    2.456
    3.5678
  intCounterNonActive=[] <empty>
  floatCounterNonActive=[] <empty>
```

Example 78 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.

```
(MeasObj=1) >show-counters -c floatCounterNonActive
floatCounterNonActive=[]
(MeasObj=1) >show-counters -v -c floatCounterActive, someOtherCounter =>
--pmJob=aJob
floatCounterActive=123.456 <PmJob=aJob> <Gp=15 min>
```

Example 79 Display Specific Counters

If no measurements are related to the DN/MO, command **show-counters** returns an error message.

```
(SystemFunctions=1) >show-counters
ERROR: Measured object for 'ManagedElement=NODE06ST
, SystemFunctions=1' does not exist
```

Example 80 Error Message for Command show-counters

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

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

```
(config-SysM=1)>ManagedElement=NODE06ST,Sy  
stemFunctions=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,S  
ystemFunctions=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=NODE06  
ST,SystemFunctions=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
, SystemFunctions=1, Fm=1
```

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

```
(config-Fm=1) >prompt $dn ManagedElement=NODE06
ST, SystemFunctions=1, Fm=1 >
```

3.4.1 Navigation Errors

The navigation error messages are described in Table 21.

Table 21 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: Unable to exit from incomplete object <current_MO_instance>	It is not possible to navigate from an MO instance that violates model constraints. The error text is followed by the problem that must be corrected.
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.



Table 21 Navigation Error Messages

Error Message	Description
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 Display MO Instances

This section describes how to display MO instances.

3.5.1 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.

3.5.2 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=NODE06ST,SystemFunctions=1,SysM=1,Snmp=1
Snmp=1
  agentAddress
    host="1.1.1.1"
    port=1111
```

Example 81 Display Single MO

3.5.3 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
Snmp=1
  agentAddress
    host="1.1.1.1"
    port=1111
  SnmpTargetV2C=1
    address="127.0.0.1"
    community="private"
```

Example 82 Display Single MO and Its Child MOs

3.5.4 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 ECLI command sequence that can be input for the ECLI. The information allows configuration data export and import by copy/paste.



```
>show-config ManagedElement=NODE06ST, SystemFunctions=1, SysM=1, Snmp=1
Snmp=1
  agentAddress
    host="1.1.1.1"
    port=1111
  up
  SnmpTargetV2C=1
    address="127.0.0.1"
    community="private"
  up
up
```

Example 83 Display Single MO and Its Child MOs in Configuration Printout Format

3.5.5 Summary of Displayed MO Properties

A summary of the elements displayed by different display options shown in Table 22.

Table 22 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



Table 22 Show MO Instance Matrix

Model Property	<code>show <path></code>	<code>show -r --recursive <path></code>	<code>show-config <path></code>	<code>show -v --verbose <path></code>	<code>show-config -v --verbose <path></code>
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
Fm=1
  heartbeatInterval=300
  lastChanged="2014-02-06T14:12:26Z" <read-only>
  lastSequenceNo=4
  sumCritical=1
  sumMajor=1
  sumMinor=0
  sumWarning=2
  totalActive=4
```

Example 84 Display MO

3.6 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 ECLI commands or session time-out.

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

- Single-valued attribute, see Section 3.6.1 Change Single-Valued Attribute on page 68
- Single-valued attribute value deletion, see Section 3.6.2 Delete Value of Single-Valued Attribute on page 70



- Sequence, see Section 3.6.3 Change Attribute Defined as Sequence on page 70
- Struct, see Section 3.6.4 Change Attribute Defined as Struct on page 82
- Passwords, see Section 3.6.5 Change Attribute Defined as Password or Passphrase String on page 83
- MO reference, see Section 3.6.6 Change Attribute Defined as MO Reference on page 85

3.6.1 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 23.

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 44, with the following exceptions:

- Strings can be entered with or without double quotation marks ("). A string including characters with a special meaning in the ECLI ([, 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 23 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



Table 23 Attribute Value Change Errors

Error Message	Semantics	Data Type
ERROR: Invalid value '<attribute_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 attribute value is incorrect and must be according to a predefined regular expression pattern. By typing <attribute_name>?, help information on the correct format to be used is displayed.	Derived 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 85 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 86 Change Single-Valued Attribute for Integer

```
(config-Snmp=1) >operationalState=DISABLED
ERROR: Attribute 'operationalState' is read-only.
```

Example 87 Change Single-Valued Attribute for Enumeration



```
(config-agentAddress)>host="$4546"  
ERROR: Invalid value '$4546' for attribute 'host'.  
Valid values are strings in a specified format.  
Type: 'host?' for more information on the format to use
```

Example 88 Change Single-Valued Attribute

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

Example 89 Change Single-Valued Attribute for EcimEmpty

3.6.2 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 24.

Table 24 Attribute Value Delete Errors

Error Message	Semantics
ERROR: Attribute <i><attribute_name></i> 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 (" <i><unknownType></i> ")	A delete operation can be performed only on attributes and object types.

```
(config-ManagedElement=NODE06ST)>show userLabel  
userLabel=[]  
(config-ManagedElement=NODE06ST)>userLabel=xyz  
(config-ManagedElement=NODE06ST)>commit -s  
(config-ManagedElement=NODE06ST)>show userLabel  
userLabel="xyz"  
(config-ManagedElement=NODE06ST)>no userLabel  
(config-ManagedElement=NODE06ST)>commit -s  
(config-ManagedElement=NODE06ST)>show userLabel  
userLabel=[]
```

Example 90 Delete Value of Single-Valued Attribute

```
(config-Snmp=1)>no operationalState  
ERROR: Attribute 'operationalState' is read-only (can't  
be deleted).
```

Example 91 Delete Value of Single-Valued Attribute

3.6.3 Change Attribute Defined as Sequence

This section describes how to change a sequence.



3.6.3.1 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
ddtStringMultivalueAttr
    "ALABAMA"
    "ALASKA"
    "ARIZONA"
```

Example 92 Initialize Sequence

3.6.3.2 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
ddtStringMultivalueAttr
    "ALABAMA"
    "ALASKA"
    "ARIZONA"
    "FLORIDA"
```

Example 93 Add Single-Valued Element to Sequence

3.6.3.3 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
ddtStringMultivalueAttr
  "ALABAMA"
  "FLORIDA"
  "ALASKA"
  "ARIZONA"
```

Example 94 Insert Simple Type Element to Sequence

3.6.3.4 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
intMultivalueAttr
  34
  42
  43
```

Example 95 Insert Simple Type Element in Sequence Using Index

For examples on related verification rules and errors, see Table 25 and Table 28.

Table 25 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.



3.6.3.5 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
ddtStringMultivalueAttr
  "ALASKA"
  "FLORIDA"
  "ARIZONA"
```

Example 96 Change Sequence Element

3.6.3.6 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
intMultivalueAttr
  34
  43
```

Example 97 Change Sequence Element Using Index



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

Example 98 Add Sequence Element Using Index

For examples on related verification rules and errors, see Table 26 and Table 28.

Table 26 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 a non-sequence attribute.

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
userLabel
  "AB"
  "Swift"
```

Example 99 Change Sequence Attribute of Type String when Value Starts with @

3.6.3.7 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
ddtStringMultivalueAttr
  "ALASKA"
  "ARIZONA"
```

Example 100 Delete Named Element from Sequence

3.6.3.8 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 101 Delete Element from Sequence Using Index

For examples on related verification rules and errors, see and Table 28.



Table 27 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.

3.6.3.9 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
ddtStringMultivalueAttr= []
```

Example 102 Delete All Elements from Sequence

3.6.3.10 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
agentAddress[@1]
  host="1.1.1.1"
  port=26343
agentAddress[@2]
  host="4.4.4.4"
  port=4444
```

Example 103 Change Element in Sequence of Struct without Key

3.6.3.11 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
(config-agentAddress[@2])>up
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
    host="0.0.0.0"
    port=26343
agentAddress[@2]
```

Example 104 Add a New Keyless Struct Element to Sequence without Index

```
(config-Snmp=1)>agentAddress[@2]
(config-agentAddress[@2])>up
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
    host="0.0.0.0"
    port=26343
agentAddress[@2]
```

Example 105 Add a New Keyless Struct Element to Sequence with Index

```
(config-Snmp=1)>agentAddress,port=999
(config-Snmp=1)>show agentAddress
agentAddress
    host="0.0.0.0"
    port=26343
agentAddress
    port=999
```

Example 106 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
agentAddress
    host="0.0.0.0"
    port=26343
agentAddress
    port=999
agentAddress
    host="9.9.9.9"
```

Example 107 Add a New Struct Element to Sequence by Setting Struct Member, with Index

```
(config-Snmp=1)>agentAddress[@5]
ERROR: Invalid index '5' for a sequence containing
1 values. Use an index <=2
```

Example 108 Add a New Keyless Struct Element to Sequence with Invalid Positional Index



3.6.3.12 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
agentAddress[@1]
  host="0.0.0.0"
  port=26222
agentAddress[@2]
  host="9.9.9.9"
  port=9999
agentAddress[@3]
  host="2.2.2.2"
  port=2222
(config-Snmp=1)>no agentAddress[@2]
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
  host="0.0.0.0"
  port=26222
agentAddress[@2]
  host="2.2.2.2"
  port=2222
```

Example 109 Delete Keyless Struct Element at Positional Index <N>

```
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
  host="0.0.0.0"
  port=26222
agentAddress[@2]
  host="9.9.9.9"
  port=9999
agentAddress[@3]
  host="2.2.2.2"
  port=2222
(config-Snmp=1)>no agentAddress[@2],host
(config-Snmp=1)>show -v agentAddress
agentAddress[@1]
  host="0.0.0.0"
  port=26222
agentAddress[@2]
  port=2222
```

Example 110 Delete Struct Member of Struct Element in Sequence of Keyless Struct

```
(config-Snmp=1) >show agentAddress
agentAddress
  host="0.0.0.0"
  port=26222
agentAddress
  host="9.9.9.9"
  port=9999
agentAddress
  host="2.2.2.2"
  port=2222
(config-Snmp=1) >no agentAddress
(config-Snmp=1) >show agentAddress
agentAddress= []
```

Example 111 Delete Whole Sequence of Keyless Struct

3.6.3.13 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[@1]
  host="0.0.0.0"
  port=26222
agentAddress[@2]
  host="9.9.9.9"
  port=9999
agentAddress[@3]
  host="2.2.2.2"
  port=2222
```

Example 112 Insert Element to Sequence of Keyless Struct

3.6.3.14 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 ECLI 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 113.



```
(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
countries="Sweden"
    capital="Stockholm"
    population=2000000
countries="Norway"
    capital="Oslo"
(config-PlainMultivalueAttrs=1) >show countries=Sweden
countries="Sweden"
    capital="Stockholm"
    population=2000000
(config-PlainMultivalueAttrs=1) >no countries=Sweden
(config-PlainMultivalueAttrs=1) >show countries
countries="Norway"
    capital="Oslo"
```

Example 113 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

3.6.3.15 Common Error Messages in Sequence Operations

The common error messages in sequence operations are shown in Table 28.

Table 28 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 <code>nonUnique</code> is not present.
ERROR: Multiplicity of the attribute (" <code><attribute_name></code> ") at max limit	The sequence element property <code>maxLength</code> is set and the number of elements is higher than this limit.



Table 28 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>.

3.6.4 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 29.

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 29 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) >productIdentity=1,p
roductDesignation=xyz
(config-ManagedElement=NODE06ST) >productIdentity=1,productNumber=1234
(config-ManagedElement=NODE06ST) >productIdentity=1,productRevision=1.1
(config-ManagedElement=NODE06ST) >commit -s
```

Example 114 Change Struct

3.6.4.1 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 3.6.3 Change Attribute Defined as Sequence on page 70.



3.6.5 Change Attribute Defined as Password or Passphrase String

Two types of passwords/passphrases exist and they are treated slightly differently in the ECLI. 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 3.6.5.1 Visible Entry of Values on page 83 and Section 3.6.5.2 Hidden Entry of Values on page 84.

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

3.6.5.1 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 ECLI `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
myPassword="ei28fwisgieatge5646i"
```

Example 115 Non-Encrypted Legacy Password

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

Example 116 Already Encrypted Legacy Password

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

Example 117 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
myPassphrase="hdg4jdldf8gfk5jd"
```

Example 118 Passphrase String, Encrypted Input and Output is Allowed

3.6.5.2

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 119.



```
(config-A=1)>myPassword=
Enter myPassword:
Repeat myPassword:
(config-A=1)>myPassword=*****
```

Example 119 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 120.

```
(config-A=1)>myPassword=
Enter myPassword:<password>
Repeat myPassword:<another_password>
ERROR: The inputs do not match
Enter myPassword:
```

Example 120 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 ECLI. However, encrypted passphrase string values can be entered at these special prompts.

3.6.6 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.
- The referred MO instance does not have to exist. If the `ManagedElement` ID does not match the one defined in the system, the ECLI automatically corrects it.



As an alternative to providing the full LDN, the ECLI 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 ECLI 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.

3.6.7 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 3.6.1 Change Single-Valued Attribute on page 68.
- Initialize sequence, see Section 3.6.3.1 Initialize Sequence on page 71.
- Add single-value element to sequence, see Section 3.6.3.2 Add Simple Type Element to Sequence on page 71.

The ECLI 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
MocA=1
  intMultiValueAttr=1
  enumMultiValueAttr
    ONE
    TWO
  stringAttr="abc"
```

Example 121 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
MocA=1
  intMultiValueAttr=1
  enumMultiValueAttr
    ONE
    TWO
  stringAttr="abc"
```

Example 122 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.

3.7 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 ECLI command for MO creation is identical to the ECLI command for changing the ECLI 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 3.6 Change MO Attributes on page 67, 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 3.7.1 MO Naming Rules on page 90) to create the desired MO.

Example of key attribute of type string:

```
(config)>ManagedElement=NODE06ST,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 ECLI 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, the ECLI position change is rejected by an error indication, which lists the names of the mandatory attributes. Example: ERROR: Following mandatory attributes are not set for the parent (NtpServer=new): <serverAddress >.
- 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 30.

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 ECLI 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 30 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) > <Tab>
address
administrativeState
community
isMibWritable
port
(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 123 Create MO

3.7.1 MO Naming Rules

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

3.7.1.1 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.



3.7.1.2 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`.

3.7.1.3 Enumerations

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

3.8 Reset MO

Command `reset` [. | `<attribute_name>`] resets either of the following to its original 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 original 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 `reset` only affects one single MO instance, it does not reset or delete any child MO instances.

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

Read-only attributes cannot be reset, as their values are assigned by the system. Key attributes and key struct members cannot be reset either. Restricted attributes can be reset 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 reset.

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

When an MO instance or an attribute is reset, the minimum multiplicity constraints of the attribute cannot be fulfilled. In this case, a warning message is displayed and navigation out of the MO instance cannot be done



until the condition is corrected. The warning messages have the format
Minimum multiplicity of `<min>` is violated for attribute
(`<attribute_name>`).

An MO instance or an attribute cannot be reset unless the current position is
inside the appropriate MO instance.

Table 31 MO Reset Errors

Error Message	Semantics
ERROR: Cannot reset key attribute <code><attribute_name></code>	A key attribute cannot be reset.
ERROR: Cannot reset read-only attribute <code><attribute_name></code>	A read-only attribute cannot be reset.
ERROR: Cannot reset restricted attribute <code><attribute_name></code>	A restricted attribute cannot be reset.
ERROR: Cannot reset the key member <code><struct_member_name></code> of the struct <code><struct_id></code>	A key member of a struct cannot be reset.
ERROR: <code><attribute_name></code> does not have any instances	A struct that does not have any instance cannot be reset.
ERROR: Cannot reset attribute, no write permission for <code><attribute_name></code>	An attribute that does not have write permission cannot be reset.

Examples

In Example 124 attribute `attrWithDefaultValue` has default value "abc" and attribute `attrWithoutDefaultValue` has been assigned value 45. Command **reset** 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)>reset
(config-TestRootMoc=1)>show -v
TestRootMoc=1
  attrWithDefaultValue="abc" <default>
  attrWithoutDefaultValue=[] <empty>
  aSimpleStruct
    memberWithDefaultValue=123 <default>
    memberWithoutDefaultValue=[] <empty>
```

Example 124 Reset MO

In Example 125 attribute `attrWithDefaultValue` has default value "abc" and the attribute is assigned value 123. Command **reset** 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)>reset attrWithDefaultValue
(config-TestRootMoc=1)>show -v attrWithDefaultValue
attrWithDefaultValue="abc" <default>
```

Example 125 Reset Attribute

In Example 126 attribute `mandatoryAttr` has minimum multiplicity 1 and no default value. Command **reset** 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)>reset mandatoryAttr
WARNING: Minimum multiplicity of 1 is violated for attribute =>
(mandatoryAttr)
(config-TestRootMoc=1)>show -v mandatoryAttr
mandatoryAttr=[] <empty>
```

Example 126 Reset Command with Minimum Multiplicity Constraints Violation

Command **reset** deletes all instances in a sequence of structs and creates one instance when only the `struct` attribute name is given as parameter, see Example 127. Otherwise, when a struct key value or an index is given, only the members of the specified instance are reset, excluding any key member, see Example 128 and Example 129.



```
(config-PlainMultivalueAttrs=1)>show -v aKeylessStruct
aKeylessStruct[@1]
  int1=[] <empty>
  str1=[] <empty>
aKeylessStruct[@2]
  int1=3
  str1=[] <empty>
(config-PlainMultivalueAttrs=1)>reset aKeylessStruct
WARNING: Minimum multiplicity of 1 is violated for attribute =>
(aKeylessStruct)
struct member: str1
WARNING: Minimum multiplicity of 1 is violated for attribute =>
(aKeylessStruct)
struct member: int1
(config-PlainMultivalueAttrs=1)>show -v aKeylessStruct
aKeylessStruct[@1]
  int1=[] <empty>
  str1=[] <empty>
```

Example 127 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)>reset 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 128 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)>reset ⇒
sequenceOfKeylessStruct [@1]
(config-ResetTestMocWithMultiValueAttr=1)>show -v ⇒
sequenceOfKeylessStruct
sequenceOfKeylessStruct [@1]
    int1=2 <default>
    str1= [] <empty>
```

Example 129 Reset a Specific Instance of a struct without a Key Member

3.9 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 32.

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 ECLI 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 32 MO Deletion Errors

Error Message	Semantics
ERROR: Cardinality is at lower limit for class (" <i><MO_class_name></i> ") , 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 ' <i><DN></i> '	The user has read permission but not delete permission.
ERROR: Command not found	The user has not read or write permission.
ERROR: Instances of ' <i><MOC_name></i> ' are not deletable	The MO cannot be deleted.

3.10 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.

3.10.1 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 name, 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 130.

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

Example 130 Action Request

3.10.1.1 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 131.

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

Example 131 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**.

3.10.2 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 33. 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 43.

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

Table 33 Action Error Messages

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



Table 33 Action Error Messages

Error Message	Semantics
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.

3.10.3 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,MOex10=1
```

2. Trigger the action, for example:



```
(config-MOex10=1)>addNumbers --num1 23 --num2 54
77
(config-MOex10=1)>concatString_defValues
comuser
(config-MOex10=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,FileM=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
```

3.11 Copy and Paste Configuration Data

The ECLI 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 ECLI 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.

3.12 Change Password Command

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

This command is available only in root position and if the ME supports changing user passwords through the ECLI. 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 132 Changing Password

Changing password can fail because of the reasons specified in Table 34.

Table 34 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 ECLI.
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.

3.13 Display ECLI Version

Command **version** displays the ECLI version.

```
>version
Ericsson CLI 1.2.0
```

Example 133 Display ECLI Version

The ECLI 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 ECLI users, including scripts and programs, about what behavior to expect from the ECLI 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 ECLI version 1.2.0.

3.13.1 Changes between ECLI Versions 1.1.0 and 1.2.0

ECLI version 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 ECLI 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.
- Option **--sort | -s** to command **show**, **show-config**, **show-mib**, and **show-table**. With this option, MO instances are sorted according to their instance names.

3.14 Pipe Utility Commands

Filtering of ECLI 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 ECLI and Subshell commands is sent as input for pipe extension commands.

3.14.1 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 35.

Table 35 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.
-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
SystemFunctions=1
Fm=1
Pm=1
```

Example 134 Filter Output of Command show Matching a Pattern

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

Example 135 Filter Output with Case-Insensitive Match

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

Example 136 Filter Output with Option -A

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

Example 137 Filter Output of Another Filter Command



```
(config-AMoc=1) >userLabel="@#$$^&* (+?>./=}" ["
(config-AMoc=1) >show
AMoc=1
  userLabel="@#$$^&* (+?>./=}" ["
  BasicThing=1
  CallableThing=1
  AThing=1
  HiddenAttrThing=1
  XThing=1
  YThing=1
(config-AMoc=1) >show | filter \@ userLabel="@#$$^&* (+?>./=}" ["
(config-AMoc=1) >show | filter \# userLabel="@#$$^&* (+?>./=}" ["
(config-AMoc=1) >show | filter \$ userLabel="@#$$^&* (+?>./=}" ["
(config-AMoc=1) >show | filter \% userLabel="@#$$^&* (+?>./=}" ["
(config-AMoc=1) >show | filter \^ userLabel="@#$$^&* (+?>./=}" ["
```

Example 138 Filter Command with Special Characters in Pattern

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 ECLI Commands Limitations

The ECLI 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.





4 Terminal Properties

This section describes the terminal properties.

4.1 Terminal Types

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

- `dumb`

The ECLI 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.

- `ossrc-eam`

The ECLI 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 ECLI for most terminal types echoes an extra space followed by a backspace as an input line reaches this length. For terminal type `ossrc-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 ECLI 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 ECLI 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 36.

Table 36 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 37.

Table 37 Delete Characters

Action	Key Combination
Delete the character before the cursor	Ctrl+H
Delete or backspace-delete the character following the cursor When the cursor is within the line, it deletes the cursor character When invoked at the end of the line, it displays all possible completions then redisplay the line When invoked on an empty line, it terminates the session	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 38.

Table 38 *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 39.

Table 39 *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 40.

Table 40 *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 41.

Table 41 *Special Actions*

Action	Key Combination
Abort a command or clear line	Ctrl+C

*Table 41 Special Actions*

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