7705 Service Aggregation Router Gen 2

Release 25.10.R1

MD-CLI User Guide

3HE 21574 AAAC TQZZA 01
Edition: 01
October 2025

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

List of tables

List of figures

1 Getting started

This guide provides information about the Model-Driven Command Line Interface (MD-CLI).

This guide is organized into functional sections and provides concepts and descriptions of the MD-CLI environment, the configuration workflow, and the syntax and command usage within the MD-CLI. It also describes how the MD-CLI interacts with the classic CLI to perform non-configuration operations.

For a list of unsupported features by platform and chassis, see the *SR OS R25.x.Rx Software Release Notes*, part number 3HE 21562 000*x* TQZZA.

Command outputs shown in this guide are examples only; actual outputs may differ depending on supported functionality and user configuration.

Note: This guide generically covers Release 25.*x*.R*x* content and may contain some content that will be released in later maintenance loads. See the *SR OS R25.x.Rx Software Release Notes*, part number 3HE 21562 000*x* TQZZA, for information about features supported in each load of the Release 25.*x*.R*x* software. For a list of features and CLI commands that are present in SR OS but not supported on the 7705 SAR Gen 2 platforms, see "SR OS Features not Supported on SAR Gen 2" in the *SR OS R25.x.Rx Software Release Notes*.

1.1 Using the MD-CLI

All references to the term “CLI” in the SR OS user documentation are generally referencing the classic CLI. The classic CLI is the CLI that has been supported in SR OS from the initial introduction of SR OS.

The MD-CLI is a management interface that can be used to manage Nokia SR OS routers. Some of the benefits of the MD-CLI include:

- follows the model-driven networking strategy, based on the same YANG models for a structured configuration and state. Consistency is maintained between the MD-CLI, NETCONF, and the gRPC model-driven interfaces.
- uses the transactional configuration method which uses a candidate configuration to hold the current configuration changes before they are applied to the running configuration, and avoids configuration ordering requirements
- provides multiuser candidate configuration modes (global, exclusive, private, and read-only) that control access to the configuration, allowing a user exclusive access to the configuration such that no other configuration changes can be made
- allows the use of configuration groups with flexible templates that simplify the configuration process by applying the template instead of repeating the same configuration
- provides MD-CLI commands to simplify integration with automation, such as displaying configuration and state in a structured format (JSON IETF or XML) and displaying contexts in an XPath format

The management interface configuration mode must be configured appropriately before using the MD-CLI. For more information, see the *7705 SAR Gen 2 System Management Guide*, “Model-Driven Management Interfaces”.

Table 1: Command syntax symbols used in this guide

Symbol	Description
|	A vertical bar represents an OR, indicating that only one of the parameters in the brackets or parentheses can be selected.
( )	Parentheses indicate that one of the parameters must be selected.
[ ]	Brackets indicate optional parameters.
Bold	Commands in **bold** indicate commands and keywords.
Italic	Commands in *italics* indicate that you must enter text based on the parameter.

In the following examples, **location** and **graceful-shutdown** are command names. For the **location** command, keyword must be one of the keywords **cf1**, **cf2**, or **cf3**. For the **graceful-shutdown** command, boolean must be one of the keywords **true** or **false**, although explicitly using the keyword **true** is optional.

location *keyword*

keyword - (**cf1** | **cf2** | **cf3**)

graceful-shutdown *boolean*

boolean - ([**true**] | **false**)

1.2 Conventions

This section describes the general conventions used in this guide.

1.2.1 Precautionary and information messages

The following information symbols are used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2.2 Options or substeps in procedures and sequential workflows

Options in a procedure or a sequential workflow are indicated by a bulleted list. In the following example, at step 1, the user must perform the described action. At step 2, the user must perform one of the listed options to complete the step.

Example: Options in a procedure

1. User must perform this step.
2. This step offers three options. User must perform one option to complete this step.
   - This is one option.
   - This is another option.
   - This is yet another option.

Substeps in a procedure or a sequential workflow are indicated by letters. In the following example, at step 1, the user must perform the described action. At step 2, the user must perform two substeps (a. and b.) to complete the step.

Example: Substeps in a procedure

1. User must perform this step.
2. User must perform all substeps to complete this action.
   a. This is one substep.
   b. This is another substep.

2 Navigating

The following sections describe navigating in the MD-CLI.

2.1 Tree structure

The MD-CLI tree contains the following elements from the Nokia YANG models:

- **container**

  This is an element that contains other elements.

Example: Containers

In the following example, **tcp-keepalive** and **gnmi** are containers.

```
tcp-keepalive {
    admin-state disable
    idle-time 600
    interval 15
    retries 4
}
gnmi {
    admin-state enable
    auto-config-save false
}
```

- **list**

  This is a sequence of list entries.

Example: Lists

In the following example, the entire set of interfaces is a list.

```
        group "group-1" {
            connect-retry 600
            keepalive 33
        }
        group "group-2" {
            description "Text description for group-2"
            local-preference 8
        }
```

- **list entry**

  This is an element similar to a container with multiple instances where each list entry is identified by the values of its keys.

Example: List entries

In the following example, group "group-2" is identified in this way.

```
router "Base" {
    bgp {
```

```
        group "group-1" {
            connect-retry 600
            keepalive 33
        }
        group "group-2" {
            description "Text description for group-2"
            local-preference 8
        }
    }
}
```

- **key**

  This is a unique identifier for a list entry.

Example: Keys

In the following example, "group-1" and "group-2" are unique identifiers.

```
router "Base" {
    bgp {
        group "group-1" {
            connect-retry 600
            keepalive 33
        }
        group "group-2" {
            description "Text description for group-2"
            local-preference 8
        }
    }
}
```

- **leaf**

  This is an element that does not contain any other elements and has a data type (for example, string or integer). A leaf can also be defined with no data type where the leaf takes no parameter value (that is, an empty leaf).

Example: Leafs

```
tcp-keepalive {
    admin-state disable
    idle-time 600
    interval 15
    retries 4
}
gnmi {
    admin-state enable
    auto-config-save false
}
```

- **leaf-list**

  This is an element that contains a sequence of values of a particular data type.

Example: Leaf-list

In the following example, "policy" is a leaf-list element.

```
    policy ["policy-a" "policy-b" "policy-c"]
```

- **leaf-list entry**

This is one of the values of a leaf-list.

Example: Leaf-list entries

In the following example, "policy-a", "policy-b", and "policy-c" are leaf-list entries.

```
    policy ["policy-a" "policy-b" "policy-c"]
```

The following terms are also used:

- **keyword**

  This is an element with a name defined by SR OS. For example, enumerated values, leaf names, and container names.

- **variable parameter**

  This is an element with a name defined by the user. For example, descriptions, names, integer or string leaf values.

- **immutable element**

  This is an element that can only be configured in the transaction in which the parent element is created. It cannot be modified while the parent element exists.

- **choice element**

  This is an element which is part of a set of mutually exclusive elements. Setting a choice element clears all configuration from the other choice elements.

Example: Elements in the tree structure

In the following example, **admin-state** (leaf name), **enable** (enumerated value), and **connect-retry** (leaf name) are keywords, and "800" is a variable parameter.

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info
    admin-state enable
    connect-retry 800
```

Managing the router configuration using the MD-CLI involves accessing and configuring the appropriate elements (containers, lists, leafs, and leaf-lists).

The MD-CLI tree shows the commands and parameters (also known as elements) that are available in a hierarchical output.

Example: tree detail command output

In the following, the bold elements are containers (or container lists) which contain leafs (or leaf-lists).

```
*[ex:/configure system]
A:admin@node-2# tree detail
+-- alarms
|   +-- admin-state <keyword>
|   +-- apply-groups <reference>
|   +-- apply-groups-exclude <reference>
|   +-- max-cleared <number>
+-- allow-boot-license-violations <boolean>
+-- apply-groups <reference>
+-- apply-groups-exclude <reference>
+-- boot-bad-exec <string>
+-- boot-good-exec <string>
```

```
+-- central-frequency-clock
|   +-- apply-groups <reference>
|   +-- apply-groups-exclude <reference>
|   +-- bits
|   |   +-- input
|   |   |   +-- admin-state <keyword>
|   |   +-- interface-type <keyword>
|   |   +-- output
|   |   |   +-- admin-state <keyword>
|   |   |   +-- line-length <keyword>
|   |   |   +-- ql-minimum <keyword>
|   |   |   +-- source <keyword>
|   |   |   +-- squelch <boolean>
|   |   +-- ql-override <keyword>
|   |   +-- ssm-bit <number>
|   +-- ptp
|   |   +-- admin-state <keyword>
|   +-- ql-minimum <keyword>
|   +-- ql-selection <boolean>
|   +-- ref-order
|   |   +-- fifth <keyword>
|   |   +-- first <keyword>
|   |   +-- fourth <keyword>
|   |   +-- second <keyword>
|   |   +-- third <keyword>

---snip---
```

2.2 Command prompt

The MD-CLI command prompt displays on two lines. The first line contains the following information:

- **baseline status indicator**

  This indicator displays an exclamation mark (!) to indicate an out-of-date baseline when in a configuration mode.

- **uncommitted changes indicator**

  This indicator displays an asterisk (*) to indicate uncommitted configuration changes when in a configuration mode.

- **configuration mode reference**

  When in a configuration mode, a configuration mode reference is displayed:

  – in round brackets for an explicit configuration workflow

  – prepended to the context, separated by a colon for an implicit configuration workflow

  The configuration mode reference can be one of the following:

  ex
  : exclusive mode

  gl
  : global mode

  pr
  : private mode

ro

read-only mode

- **context**

  The present working context is displayed in square brackets ([]) when in operational or configuration mode.

For an explicit configuration workflow, the format of the first line is as follows:

<*baseline status indicator* > <*uncommitted changes indicator*> (<*configuration mode*>) [*context*]

Example: Explicit configuration workflow

```
(ro)[/]

(ex)[/configure router "Base" bgp]
```

For an implicit configuration workflow, the format of the first line is as follows:

<*baseline status indicator*> <*uncommitted changes indicator*> [<*configuration mode*>:*context*]

Example: Implicit configuration workflow

```
[ro:/configure]

*[ex:/configure]
```

The second line contains the following information:

- **user**

  The user is the name of the current user for this session.

- **name**

  The name is the system name, as configured with the **configure system name** command. The system name can change dynamically during the session if it is configured to a different name.

The format of the second line is as follows:

CPM:*user@name#*

Example: Two-line prompt usage

The following examples display the two-line prompt in different modes.

- prompt in operational mode

  ```
  [/]
  A:admin@node-2#
  ```

- prompt in the operational root, with exclusive configuration mode

  ```
  (ex)[/]
  A:admin@node-2#
  ```

- prompt in operational mode **show router bgp**

  ```
  [/show router "Base" bgp]
  A:admin@node-2#
  ```

- prompt in exclusive configuration mode **configure router bgp**

```
(ex)[/configure router "Base" bgp]
A:admin@node-2#
```

- prompt in exclusive configuration mode **configure router bgp** with uncommitted changes

```
*(ex)[/configure router "Base" bgp]
A:admin@node-2#
```

- implicit configuration workflow prompt for a session in private configuration mode, with a present working context of **configure router bgp** with uncommitted changes in the private candidate datastore, and the baseline datastore out-of-date

```
!*[pr:/configure router "Base" bgp]
A:admin@node-2#
```

2.3 Environment commands

The MD-CLI has two environment configurations:

- the global environment configuration in the **configure system management-interface cli md-cli** context, which is persistent in the saved configuration file. Changes made to the global environment configuration apply only to new sessions and do not affect current sessions.
- the per-session environment configuration in the **environment** context, which is not persistent and applies only to each user's current session. The global environment configuration is copied to the per-session environment configuration when a new session is started.

See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for information about the **environment** commands in the MD-CLI.

2.3.1 Command aliases

The MD-CLI can be customized to provide an enhanced user experience through the use of command aliases.

Aliases can be configured to execute a user-defined command name that performs a specific action. Supported actions include:

- executing an MD-CLI command
- executing a Python application

2.3.1.1 Creating aliases

Aliases can be created and modified in the per-session context or the global configuration **configure system management-interface cli md-cli environment** context. Aliases (or elements of the aliases) that are deleted in the per-session **environment** context return to their defaults (usually to the state of the global environment alias defined in the configuration context).

When an alias is configured, the alias acts as a substitute for a command or set of commands that can then be executed. Nokia recommends defining alias names that are meaningful to the user. Alias names must not use any string that is an MD-CLI root element, such as **admin**, or that is an MD-CLI global command, such as **insert**. Aliases are displayed in command completion and ? help, where applicable.

The MD-CLI command history displays what was entered in the CLI. Instead of recording the resultant operation, the history records the alias name and any arguments.

Typical use cases for MD-CLI command aliases include:

- shortcuts for navigating MD-CLI contexts
- creating new MD-CLI output commands
- aliasing existing MD-CLI commands

Note: Expressions (wildcard, range, and regular expressions) are not supported in alias list keys.

2.3.1.2 Mounting an alias

Aliases are mounted at specific points in the MD-CLI contexts. Aliases may be mounted globally using the **mount-point global** option. A globally mounted alias may be executed from any MD-CLI context.

Example: Execution of a globally mounted alias

The following example shows the execution of a globally mounted alias that creates the alias **up** for the **back** command.

```
command-alias {
    alias "up" {
        admin-state enable
        cli-command "back"
        mount-point global { }
    }
}

[ex:/configure router "Base" bgp]
A:admin@node-2# up  # Moved back one level by typing “up”


[ex:/configure router "Base"]
A:admin@node-2# uPress Tab  # Integrated into MD-CLI command completion

 up

 update

[ex:/configure router "Base"]
A:admin@node-2# up ?  # Accepts parameters and displays ? help for aliased command

 [[levels] <number>]
 <number> - <1..4294967295>
 Default  - 1

    Number of levels to move up
```

Alternatively, an alias may be mounted at one or more specific locations, which limits the contexts from which the alias may be executed. The execution of such an alias can be considered equivalent to executing the resultant operation from the present working context of the mount point.

Example: Execution of an alias mounted in the /show and /tools perform contexts

```
command-alias {
    alias "hello-from-some-places-only" {
        admin-state enable
        python-script "hello"
        mount-point "/show" { }
        mount-point "/tools perform" { }
    }
}

[/]
A:admin@node-2# show hello-from-some-places-only
Hi

[/]
A:admin@node-2# tools perform hello-from-some-places-only
Hi

[/show]
A:admin@node-2# hello-from-some-places-only
Hi

[/tools]
A:admin@node-2# hello-from-some-places-only
               ^^^^^^^^^^^^^^^^^^^^^^^^^^
MINOR: MGMT_CORE #2201: Unknown element - 'hello-from-some-places-only'
```

Mount points may contain lists. Where lists are provisioned, the keys must be provided. If lists have default keys, these may be omitted. List keys do not need to be present in the current configuration.

Example: Alias using a nonexistent list key in a mount point

The key *vpn3* does not exist as a current VPRN service. When *vpn3* is entered on the command line, the resultant configuration shows the string *vpn3* enclosed in quotation marks, escaped by the backslash (\) character.

```
[ex:/configure system management-interface cli md-cli environment command-alias]
A:admin@node-2# info /state service vprn Press Tab

 <service-name>
 "vpn1"
 "vpn2"

[ex:/configure system management-interface cli md-cli environment command-alias alias
 "list-keys-in-mount-point-example"]
A:admin@node-2# mount-point "/state service vprn vpn3"

*[ex:/configure system management-interface cli md-cli environment command-alias alias
 "list-keys-in-mount-point-example"]
A:admin@node-2# info
    admin-state enable
    python-script "hello"
    mount-point "/state service vprn \"vpn3\"" { }
```

Example: Alias automatically available

When the list key is created, the alias automatically becomes available in the context, as shown in the following example.

```
[/]
A:admin@node-2# state service vprn "vpn3" list-keys-in-mount-point-example
                                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
MINOR: MGMT_CORE #2201: Unknown element - 'list-keys-in-mount-point-example'

[/]
A:admin@node-2# edit-config exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

(ex)[/]
A:admin@node-2# configure service vprn vpn3

*(ex)[/configure service vprn "vpn3"]
A:admin@node-2# customer "1"

*(ex)[/configure service vprn "vpn3"]
A:admin@node-2# commit

(ex)[/configure service vprn "vpn3"]
A:admin@node-2# state service vprn vpn3 list-keys-in-mount-point-example
```

Note: Ranges are not supported in **cli-command** or **mount-point** commands.

2.3.1.3 Configuring aliases to execute Python applications

An alias can execute a configured Python application. The **alias python-script** command can reference a Python application configured in the **/configure python python-script** context.

See Accounting and authorization of aliases for information about authorizing Python application execution using the **alias** command.

2.3.1.4 Configuring aliases to execute MD-CLI commands

An alias may execute an MD-CLI command by using the **alias cli-command** command. The command is configured using a string that contains the MD-CLI command to execute. The string must be syntactically correct and quotation marks (single or double quotes) must be escaped using the backslash (\) character. If the string contains a list, the list key must be provided, except in the following cases:

- the list key has a default entry
- the list key is the last element of the CLI command

Example: Simplified navigation within the MD-CLI

```
command-alias {
    alias "go-to-alias" {
        admin-state enable
        cli-command "configure system management-interface cli md-cli environment command-
alias"
        mount-point global { }
```

```
    }
}

[ex:/configure router "Base" bgp]
A:admin@node-2# go-to-alias

[ex:/configure system management-interface cli md-cli environment command-alias]
A:admin@node-2#
```

Example: List (router) with a default list key ("Base")

```
[/state system]
A:admin@node-2# show router-auth-stats

===============================================================================
Authentication Global Statistics
===============================================================================
Client Packets Authenticate Fail     : 0
Client Packets Authenticate Ok       : 0
===============================================================================
```

Example: Key provided as a parameter

A list key can be omitted if the last element of the CLI command string is a list. This allows the key to be provided as a parameter when executing the alias, as shown in the following example.

```
command-alias {
    alias "vprn-state" {
        admin-state enable
        cli-command "info candidate /state service vprn"
        mount-point "/show" { }
    }
}

[/]
A:admin@node-2# show vprn-state
                     ^^^^^^^^^^
MINOR: CLI #2001: Missing element value - 'service-name'

[/]
A:admin@node-2# show vprn-state vpn1
    oper-service-id 1
    oper-state up
    sap-count 2
    sdp-bind-count 0
    template-used ""
    creation-origin manual
    vrtr-id 2
    oper-router-id 255.0.0.0
    oper-route-distinguisher-type auto

...snip...
```

Example: Extra parameters executed

Aliases can also accept arguments, allowing extra parameters to be executed, as shown in the following example.

```
[/]
A:admin@node-2# show vprn-state vpn1 interface "test1" ipv4 icmp
    statistics {
```

```
        icmp-in-msgs 0
        icmp-in-errors 0
        icmp-in-dest-unreachables 0
        icmp-in-redirects 0
        icmp-in-echos 0
        icmp-in-echo-replies 0
        icmp-in-time-exceeds 0
        icmp-in-src-quenches 0
        icmp-in-timestamps 0
        icmp-in-timestamp-replies 0
        icmp-in-address-masks 0
        icmp-in-address-mask-replies 0
        icmp-in-parm-problems 0
        icmp-out-msgs 0
        icmp-out-errors 0
        icmp-out-dest-unreachables 0
        icmp-out-redirects 0
        icmp-out-echos 0
        icmp-out-echo-replies 0
        icmp-out-time-exceeds 0
        icmp-out-src-quenches 0
        icmp-out-timestamps 0
        icmp-out-timestamp-replies 0
        icmp-out-address-masks 0
        icmp-out-address-mask-replies 0
        icmp-out-parm-problems 0
        icmp-out-discards 0
    }
```

Example: Include output modifiers in definitions

Aliases that execute CLI commands may also include output modifiers in the definitions. In the following example, the alias **bgp-top-line** outputs the line from **show router bgp summary** that uses the output match modifier to select a specific string:

```
command-alias {
    alias "bgp-top-line" {
        admin-state enable
        cli-command "show router bgp summary | match \"AS:\""
        mount-point global { }
    }
}

[/]
A:admin@node-2# bgp-top-line
 BGP Router ID:255.0.0.0        AS:65535       Local AS:65535
```

2.3.1.5 Accounting and authorization of aliases

Aliases have specific accounting and authorization rules, as shown in the following configuration example.

Example

```
command-alias {
    alias "my-python-alias" {
        admin-state enable
        python-script "hello"
        mount-point global { }
    }
    alias "my-cli-alias" {
```

```
        admin-state enable
        cli-command "info running /state service vprn"
        mount-point global { }
    }
    alias "my-cli-python-alias" {
        admin-state enable
        cli-command "pyexec cf3:\hello.py"
        mount-point global { }
    }
}
```

The command entered by the user on the MD-CLI command line is sent to the accounting system, including all arguments supplied to the alias.

For example, if the user enters **my-cli-alias vpn1**, the string *my-cli-alias vpn1* is sent to the accounting system. If the user enters **my-python-alias**, the string *my-python-alias* is sent to the accounting system.

The string sent to the authorization system depends on whether **cli-command** or **python-script** is configured in the **alias** context.

When an alias uses a CLI command, the resulting MD-CLI command is sent to the authorization system. For example, if the user enters **my-cli-alias vpn1**, the string sent to the authorization system is *info running /state service vprn vpn1*. If the user enters **my-cli-python-alias**, the string sent to the authorization system is *pyexec cf3:\hello.py*

If the alias is configured with the **python-script** option, the string sent to the authorization server is the referenced Python application name prefaced by the string *python-script*. This allows Python applications initiated with the **pyexec** command to be authorized independently of those initiated from within MD-CLI command aliases. Administrators can predefine approved Python applications that may be executed by users who otherwise have no access to run the **pyexec** command.

For example, if the operator enters **my-python-alias**, the string sent to the authorization server is *python-script \"hello\"*.

2.3.2 Customizing per-session environment settings

The environment can be customized for all sessions in the configuration under the **configure system management-interface cli md-cli environment** context, or per session using the **environment** command. When a new MD-CLI session is started, the per-session environment configuration is copied from the global environment configuration. Changes made to the global environment configuration after the session begins apply only to new sessions and do not affect current sessions. Changes made to the environment parameters for a session apply only for that session.

The per-session environment is accessed by entering **environment** at the operational root or with **/ environment** from any other mode or context. Changes made in the per-session environment are immediate.

The **info** command displays the difference between the per-session environment and the configured global environment parameters. Therefore, for a new MD-CLI session, the **info** command has no output, as the per-session environment is the same as the global environment. The **info detail** command displays the current values in the global environment for all parameters.

2.3.3 Customizing the session prompt

2.3.3.1 Customizing the uncommitted changes indicator

As the default setting of the environment configuration, the uncommitted changes indicator is displayed as part of the command prompt. This setting can be modified per session or it can be changed for all MD-CLI sessions by changing the environment configuration.

The **uncommitted-changes-indicator** command under the **environment prompt** context suppresses or displays the change indicator for an MD-CLI session. Environment changes are applied immediately and are lost when the session disconnects.

Example

```
*[/environment prompt]
A:admin@node-2# uncommitted-changes-indicator false

[/environment prompt]
A:admin@node-2#

[/environment prompt]
A:admin@node-2# uncommitted-changes-indicator true

*[/environment prompt]
A:admin@node-2#
```

2.3.3.2 Customizing the line preceding the command prompt

By default, a blank line precedes the command prompt. This setting can be modified for each MD-CLI session.

The **newline** command under the **environment prompt** context suppresses or displays a new line before the prompt.

Example

```
[/]
A:admin@node-2# environment prompt

[/environment prompt]
A:admin@node-2# newline false
[/environment prompt]
A:admin@node-2# newline true

[/environment prompt]
A:admin@node-2#
```

2.3.3.3 Customizing the context information in the command prompt

By default, the context is displayed in the command prompt. This setting can be modified for each MD-CLI session.

The **context** command under the **environment prompt** context suppresses or displays the current context.

Example

```
[/environment prompt]
A:admin@node-2# context false

[]
A:admin@node-2# context true

[/environment prompt]
A:admin@node-2#
```

2.3.3.4 Customizing the date and time output

By default, the timestamp is not displayed before the command prompt. This setting can be modified for each MD-CLI session.

The **timestamp** command under the **environment prompt** context suppresses or displays the timestamp.

Example: Suppress or display the timestamp

```
[/environment prompt]
A:admin@node-2# timestamp true

Tue, 16 Feb 2021 16:37:26 UTC
[/environment prompt]
A:admin@node-2# timestamp false

[/environment prompt]
A:admin@node-2#
```

The **environment time-display** command configures the time zone display to UTC or local time (as configured in **configure system time**).

Example: Configure the time zone display

```
[/environment]
A:admin@node-2# time-display ?

 time-display <keyword>
 <keyword> - (local|utc)
 Default   - local


    Time zone to display time
```

The **environment time-format** command specifies the format for the time display.

Example: Specify the format for the time display

```
[/environment]
A:admin@node-2# time-format ?

 time-format <keyword>
 <keyword> - (iso-8601|rfc-1123|rfc-3339)
 Default   - rfc-3339
```

```
    Format to display the date and time
```

Example: Show the time in the format defined by ISO 8601

```
[/state cpm "a" hardware-data]
A:admin@node-2# software-last-boot-time
    software-last-boot-time "2020-09-01 23:27:17 UTC"
```

Example: Show the time in the format defined by RFC 1123

```
[/state cpm "a" hardware-data]
A:admin@node-2# software-last-boot-time
    software-last-boot-time "Tue, 01 Sep 2020 23:27:17 UTC"
```

Example: Show the time in the format defined by RFC 3339

```
[/state cpm "a" hardware-data]
A:admin@node-2# software-last-boot-time
    software-last-boot-time 2020-09-01T23:27:17.0+00:00
```

2.3.4 Customizing the progress indicator

The progress indicator appears on the line immediately following the command and disappears when the MD-CLI command completes or when output is available to display. The indicator is a display of dynamically changing dots.

Example: Progress indicator

```
[ex:/configure]
A:admin@node-2# compare
...                   # progress indicator displays here as dots
```

The delay interval can be configured using the **delay** command or the indicator can be disabled with the **admin-state disable** command under the **environment progress-indicator** context.

Example: Customizing the progress indicator for logged sessions

```
[/environment progress-indicator]
A:admin@node-2# ?

 admin-state          - Administrative state of the progress indicator
 delay                - Delay before progress indicator is displayed
 type                 - Progress indicator output style
```

2.3.5 Customizing the pagination setting

The **environment more** command enables pagination when configured to **true** and disables pagination when configured to **false**.

Example: Using the environment more command

```
[/]
```

```
A:admin@node-2# environment more true
```

```
[/]
A:admin@node-2# show system security management

===============================================================================
Server Global
===============================================================================
Telnet:
Administrative State         : Enabled
Operational State            : Up
Telnet6:
Administrative State         : Disabled
Operational State            : Down
FTP:
Administrative State         : Disabled
Operational State            : Down
SSH:
Administrative State         : Enabled
Operational State            : Up
NETCONF:
Administrative State         : Disabled
Operational State            : Down
GRPC:
Administrative State         : Disabled
Operational State            : Down

--(more)--(39%)--(lines 1-23/55)--
```

The pagination setting can be overridden by using **| no-more** for a single command. As with pagination disabled, the output is displayed completely without any prompts to continue.

Example: Overriding the pagination setting

```
[/]
A:admin@node-2# show system security management | no-more
```

2.3.6 Customizing the console settings

The default size for a console terminal is 80 characters wide by 24 lines long. The **environment console** command can be used to change these settings.

Example

```
[/environment]
A:admin@node-2# console ?

 console

 length               - Number of lines displayed on the screen
 width                - Number of columns displayed on the screen
```

2.3.7 Customizing the message level security settings

The INFO: CLI messages are displayed by default. The **environment message-security-level** command suppresses the INFO messages by changing the setting to **warning**.

Example

```
[/environment message-severity-level]
A:admin@node-2# cli ?

 cli <keyword>
 <keyword> - (warning|info)
 Default   - info

    Message severity threshold for CLI messages
```

Example

Following are examples of INFO: CLI messages that are suppressed when the setting is changed to **warning**.

```
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2052: Switching to the MD-CLI engine
INFO: CLI #2054: Entering global configuration mode
INFO: CLI #2056: Exiting global configuration mode
INFO: CLI #2055: Uncommitted changes are present in the candidate configuration
INFO: CLI #2057: Uncommitted changes are kept in the candidate configuration
```

2.3.8 Customizing configuration output display

Default values are always displayed in the **info detail** command output and not in the **info** command output. Use the **environment info-output always-display admin-state** command to display the values of the **admin-state** element in the **info** command output (without using the **detail** option), even if these values are set to the default. Enabling the **admin-state** command allows the **info** command output to always show the **admin-state** value without showing all the configured, default, and unconfigured values that are displayed using the info **detail** command. For more information, see Using the info command.

Example

In the following configuration output, the **admin-state** for port 1/1/2 with a default configuration is not displayed using the **info** command. The **admin-state** is displayed after the **environment info-output always-display admin-state** command is configured to **true**.

```
[ex:/configure port 1/1/2]
A:admin@node-2# info

[ex:/configure port 1/1/2]
A:admin@node-2# /environment info-output always-display admin-state

[ex:/configure port 1/1/2]
A:admin@node-2# info
    admin-state disable
```

2.3.9 Preventing changes to environment settings

The environment datastore is subject to AAA command authorization. A user can be prevented from modifying the global environment settings or the per-session environment settings, or both.

Example

In the following configuration output, **entry 113** blocks user "tstuser" from modifying the global environment settings. In addition, **entry 114** prevents the user from changing the per-session environment settings.

```
(ro)[/configure system security aaa local-profiles profile "tstuser"]
A:admin@node-2# info
    default-action permit-all
    entry 113 {
        action deny
        match "configure system management-interface cli md-cli environment"
    }
    entry 114 {
        action deny
        match "environment"
    }
```

```
[ex:/configure system management-interface cli md-cli environment]
A:tstuser@node-2# prompt timestamp
MINOR: MGMT_CORE #2020: Permission denied

[ex:configure system management-interface cli md-cli environment]
A:tstuser@node-2# /environment
MINOR: MGMT_CORE #2020: Permission denied

[ex:/configure system management-interface cli md-cli environment]
A:tstuser@node-2#
```

2.4 Using online help

A short help description is displayed immediately when the question mark (?) is entered (without needing to press Enter).

Example: Help displayed from the operational root level

```
[/]
A:admin@node-2# ?

 admin                 + Enter the administrative context for system operations
 bof                   + Enter the bof context
 clear                 + Clear statistics or reset operational state
 configure             + Enter the configuration context
 debug                 + Enter the debug context
 environment           + Configure the environment settings for this session
 file                  + Perform file operations
 monitor               + Show operational information at an interval
 password              - Change the local user or an administrative password
 show                  + Show operational information
 state                 + Show state information
 tools                 + Enter the tools context for troubleshooting

 Global commands:
 back                  - Move back one or more levels
 delete                - Delete an element
 edit-config           - Enter a configuration context
 enable                - Enable administrative mode
 exec                  - Execute commands from a file
```

```
 exit                  - Return to the previous context or to operational root
 history               - Show the command history
 info                  - Show the information under the present working context
 logout                - Exit the CLI session
 oam                   - Perform OAM tests
 ping                  - Ping an IP address or DNS name
 pwc                   - Show the present working context
 pyexec                - Execute a Python application
 ssh                   - SSH to an IP address or DNS name
 telnet                - Telnet to an IP address or DNS name
 top                   - Move to the top level of the context
 traceroute            - Show the route taken to an IP address or DNS name
 tree                  - Show the command tree under the present working context
```

Example: Help displayed in exclusive configuration mode

? help output lists additional commands available in exclusive configuration mode. The ? help is context-sensitive.

```
(ex)[/]
A:admin@node-2# ?

 admin                 + Enter the administrative context for system operations
 clear                 + Clear statistics or reset operational state
 configure             + Enter the configuration context
 environment           + Configure the environment settings for this session
 file                  + Perform file operations
 monitor               + Show operational information at an interval
 password              - Change the local user or an administrative password
 show                  + Show operational information
 state                 + Show state information
 tools                 + Enter the tools context for troubleshooting

 Global commands:
 back                  - Move back one or more levels
 delete                - Delete an element
 edit-config           - Enter a configuration context
 enable                - Enable administrative mode
 exec                  - Execute commands from a file
 exit                  - Return to the previous context or to operational root
 history               - Show the command history
 info                  - Show the information under the present working context
 insert                - Insert an element into a user-ordered list
 logout                - Exit the CLI session
 oam                   - Perform OAM tests
 ping                  - Ping an IP address or DNS name
 pwc                   - Show the present working context
 pyexec                - Execute a Python application
 quit-config           - Exit the candidate configuration mode
 ssh                   - SSH to an IP address or DNS name
 telnet                - Telnet to an IP address or DNS name
 top                   - Move to the top level of the context
 traceroute            - Show the route taken to an IP address or DNS name
 tree                  - Show the command tree under the present working context

 Configuration commands:
 annotate              - Annotate a configuration element with a comment
 commit                - Commit the candidate configuration
 compare               - Compare changes between datastores
 copy                  - Copy a configuration element to another
 discard               - Discard changes in the candidate datastore
 rename                - Rename a list element
 update                - Update the candidate baseline
```

```
 validate              - Validate changes in the candidate datastore
```

Example: Help results depend on cursor position

The help results may depend on the cursor position. The following example shows the **router** command syntax, followed by available commands after entering the **router** context.

```
[ex:/configure]
A:admin@node-2# router?

 router [[router-name] <string>]

 [router-name]         - Administrative router name

 aggregates            + Enter the aggregates context
 allow-icmp-redirect   - Allow ICMP redirects on the management interface
 allow-icmp6-redirect  - Allow IPv6 ICMP redirects on the management interface
 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 autonomous-system     - AS number advertised to peers for this router
 bfd                   + Enter the bfd context
 bgp                   + Enable the bgp context
 bier                  + Enable the bier context
 class-forwarding      - Allow class-based forwarding over IGP shortcuts
 confederation         + Enter the confederation context
 description           - Text description
 dhcp-server           + Enter the dhcp-server context
 dns                   + Enter the dns context
 ecmp                  - Maximum equal-cost routes for routing table instance

---snip---
```

Example: ? output with more detail

In the following ? output, similar information is shown, with more details provided for configuring the **router** command, including the allowable string length and default value for the command.

```
[ex:/configure]
A:admin@node-2# router ?

 router [[router-name] <string>]

 [[router-name] <string>]
 <string> - <1..64 characters>
 Default  - "Base"

    Administrative router name

 aggregates            + Enter the aggregates context
 allow-icmp-redirect   - Allow ICMP redirects on the management interface
 allow-icmp6-redirect  - Allow IPv6 ICMP redirects on the management interface
 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 autonomous-system     - AS number advertised to peers for this router
 bfd                   + Enter the bfd context
 bgp                   + Enable the bgp context
 bier                  + Enable the bier context
 class-forwarding      - Allow class-based forwarding over IGP shortcuts
 confederation         + Enter the confederation context
 description           - Text description
 dhcp-server           + Enter the dhcp-server context
 dns                   + Enter the dns context
```

```
 ecmp                 - Maximum equal-cost routes for routing table instance

 ---snip---
```

2.4.1 Indicators in the online help

The following table describes the indicators available in the online help.

Table 2: Indicators in the online help

Symbol	Description
+	Indicates a container or list
-	Indicates a leaf, a leaf-list, a list or container with no leafs, or a global command (if in the operational root)
^	Indicates a mandatory element (an element that must be configured before the configuration is considered valid)
:	Indicates the first element of a group of choice elements that are mutually exclusive

Example

In the following help display example, the containers are **eth-cfm**, **domain**, and **association**. The leafs are **apply-groups**, **apply-groups-exclude**, **dns**, **format**, **level**, **mac**, **md-index**, and **name**, while **level** is also a mandatory element.

```
[ex:/configure]
A:admin@node-2# eth-cfm ?

 eth-cfm

 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 default-domain        + Enter the default-domain context
 domain                + Enter the domain list instance
```

```
[ex:/configure eth-cfm]
A:admin@node-2# domain ?

 [md-admin-name] <string>
 <string> - <1..64 characters>

    Unique domain name
```

```
[ex:/configure eth-cfm]
A:admin@node-2# domain dom-name ?

 domain

 Immutable fields      - level, dns, mac, name, format, md-index

 apply-groups          - Apply a configuration group at this level
```

```
apply-groups-exclude  - Exclude a configuration group at this level
association           + Enter the association list instance
level                 ^ Maintenance Domain Level (MD Level)
md-index              - The index of the Maintenance Domain (MD)

Mandatory choice: md-name
dns                  :- Domain name like text string derived from a DNS name
format               :- Maintenance domain name not to be provided
mac                  :- Maintenance domain MAC name
name                 :- Maintenance domain name as an ASCII string
```

2.4.1.1 Descriptions and format guidelines for leafs and leaf-lists

When online help is entered for a leaf or leaf-list, a short description of the element is displayed after the element type. The valid input values for the element are also listed, as shown in the following examples.

The **description** string for the VPRN service can have a length of 1 to 80 characters.

Example: description string for a VPRN service

```
*[ex:/configure service vprn "5"]
A:admin@node-2# description ?

 description <string>
 <string> - <1..80 characters>

    Text description
```

The ? help for the **autonomous-system** command lists the valid number range, followed by a short description of the parameter.

Example: autonomous-system command with the valid number range

```
*[ex:/configure service vprn "5"]
A:admin@node-2# autonomous-system ?

 autonomous-system <number>
 <number> - <1..4294967295>

    AS number advertised to peers for this router
```

A parameter value may have a unit type associated with it.

Example: ingress-buffer-allocation command with an associated unit type

```
*[ex:/configure qos sap-ingress "sap-pname" policer 6]
A:admin@node-2# mbs ?

 mbs (<number> | <keyword>)
 <number>  - <0..268435456> - bytes
 <keyword> - auto           - bytes
 Default   - auto

    High priority violate threshold of PIR leaky bucket
```

The **owner** command refers to the script policy name that is configured through the **configure system script-control script-policy** context. The name is a string of 1 to 32 characters.

Example: Parameter that is a reference to another parameter

```
*[ex:/configure log event-handling handler "h-name" entry 5]
A:admin@node-2# script-policy owner ?

 owner <reference>
 <reference> - <1..32            - configure system script-control script-
               characters>         policy <./name> owner <owner>
 Default     - TiMOS CLI

    Script policy owner
```

2.4.1.2 Immutable elements

An immutable element can only be configured in the transaction in which the parent element is created. It cannot be modified while the parent element exists. Any modification to an immutable element in model-driven interfaces causes SR OS to automatically delete the parent element and recreate it with the new value for the immutable element.

Example: Immutable elements defined in online help

```
[ex:/configure eth-cfm]
A:admin@node-2# domain example_domain ?

 domain

 Immutable fields      - level, dns, mac, name, format, md-index

 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 association           + Enter the association list instance
 level                 ^ Maintenance Domain Level (MD Level)
 md-index              - The index of the Maintenance Domain (MD)

 Mandatory choice: md-name
 dns                  :- Domain name like text string derived from a DNS name
 format               :- Maintenance domain name not to be provided
 mac                  :- Maintenance domain MAC name
 name                 :- Maintenance domain name as an ASCII string
```

```
[ex:/configure eth-cfm]
A:admin@node-2# domain example_domain level ?

  level <number>
 <number> - <0..7>

 'level' is: mandatory, immutable

    Maintenance Domain Level (MD Level)

    Warning: Modifying this element recreates
    'configure eth-cfm domain "example_domain"' automatically for the new value
    to take effect.
```

2.4.1.3 Mutually exclusive choice elements

Elements that are part of a choice are listed in a separate section in the online help. Mandatory choices are listed first. Each choice contains a set of mutually exclusive elements or groups of elements. The first element of a group is indicated with a colon (:).

Example

The following example shows a set of two mutually exclusive choice elements for an ingress queue rate. If configuring one of the choice elements, either the **cir** and **fir** values can be configured or the **police** value.

```
*[ex:/configure qos sap-ingress "ing-1" queue 1 rate]
A:admin@node-2# ?

 pir                   - Administrative PIR

 Choice: rate-cir-fir-or-police
 cir                  :- Administrative CIR
 fir                   - Administrative FIR
 police               :- Drop the traffic feeding into queue above the PIR rate
```

2.4.1.4 Optional indicators in the online help

The following help display is an example of optional indicators.

Example: Optional indicators

The square brackets ([]) around **slot-number** indicate that the **slot-number** keyword is optional when entering the command.

```
[ex:/configure]
A:admin@node-2# card ?

 [slot-number] <number>
 <number> - <1..20>

    Slot number within the chassis
```

The **card** context can be entered as one of the following.

```
[ex:/configure]
A:admin@node-2# card slot-number 5
```

or

```
[ex:/configure]
A:admin@node-2# card 5
```

Angle brackets (<>) indicate a variable name and the vertical bar (|) indicates a choice.

Example: sub-group command

For the **sub-group** command, a number in the range of 1 to 8 can be entered, or one of the keywords **auto-iom** or **auto-mda**.

```
*[ex:/configure lag "lag-8" port 1/1/1]
A:admin@node-2# sub-group ?

 sub-group (<number> | <keyword>)
 <number>  - <1..8>
 <keyword> - (auto-iom|auto-mda)
 Default   - 1

 'sub-group' is: immutable

    Subgroup of the port in the LAG

    Warning: Modifying this element recreates
    'configure lag "lag-8" port 1/1/1' automatically for the new value to take
    effect.
```

2.5 Available commands

See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for more information about all available commands.

See Editing configuration in this guide for more information about the configuration commands.

The following commands are available at the operational root level of the MD-CLI hierarchy.

```
[/]
A:admin@node-2# ?

 bof                   + Enter the bof context
 configure             + Enter the configure context
 debug                 + Enter the debug context
 environment           + Enter the environment context
```

Global commands are available from various levels of the MD-CLI hierarchy.

```

 Global commands:
 admin                 + Enter the administrative context for system operations
 back                  - Move back one or more levels
 clear                 + Clear statistics or reset operational state
 delete                - Delete an element
 edit-config           - Enter a configuration mode
 enable                - Enable administrative mode
 exec                  - Execute commands from a file
 exit                  - Return to the previous context or to operational root
 file                  + Perform file operations
 history               - Show the command history
 info                  - Show the information under the present working context
 logout                - Exit the CLI session
 monitor               + Show operational information at an interval
 oam                   - Perform OAM tests
 password              - Change the local user password
 perform               + Perform troubleshooting and debugging
 ping                  - Ping an IP address or DNS name
```

```
pwc                   - Show the present working context
pyexec                - Execute a Python application
reset                 + Clear statistics or reset operational state
show                  + Show operational information
ssh                   - SSH to an IP address or DNS name
state                 + Show state information
telnet                - Telnet to an IP address or DNS name
tools                 + Perform troubleshooting and debugging
top                   - Move to the top level of the context
traceroute            - Show the route taken to an IP address or DNS name
tree                  - Show the command tree under the present working context
```

Configuration commands are available within a configuration mode. However, some commands are not visible within specific configuration modes; for example, the configuration commands available in read-only configuration mode are limited.

Example: configuration commands in read-only configuration mode

```
compare               - Compare changes between datastores
validate              - Validate changes in the candidate datastore
```

Example: configuration commands in exclusive configuration mode

```
annotate              - Annotate a configuration element with a comment
commit                - Commit the candidate configuration
compare               - Compare changes between datastores
copy                  - Copy a configuration element to another
discard               - Discard changes in the candidate configuration
load                  - Load contents into the candidate configuration
rename                - Rename a list element
replace               - Match and replace values in the candidate configuration
rollback              - Roll back to a previous configuration
update                - Update the candidate baseline
validate              - Validate changes in the candidate configuration
```

2.6 Navigating hierarchy levels

The following commands can be used to navigate the MD-CLI hierarchy (context) levels:

- **pwc**

  The **pwc** command displays the present working context with all keyword and variable parameters.

```
(ex)[/]
A:admin@node-2# configure

(ex)[/configure]
A:admin@node-2# card 1

(ex)[/configure card 1]
A:admin@node-2# mda 2

*(ex)[/configure card 1 mda 2]
A:admin@node-2# network

*(ex)[/configure card 1 mda 2 network]
```

```
A:admin@node-2# pwc
Present Working Context:
  configure
  card 1
  mda 2
  network
```

– **pwc previous**

  The **pwc previous** command displays the previous working context.

```
*[ex:/configure card 1 mda 2 network]
A:admin@node-2# pwc previous
Previous Working Context:
  configure
  card 1
  mda 2
```

– **pwc path-type**

  The **pwc** command has several options to display the path in different formats.

  - **model-path**

    Displays a YANG-modeled format that can be used with RESTCONF-based management systems.

  - **gnmi-path**

    Displays a format that can be used with gNMI streaming telemetry.

  - **cli-path**

    Displays a single-line version of the MD-CLI format that can be copied and pasted into an MD-CLI command with a CLI path as input.

  - **json-instance-path**

    Displays a YANG-modeled path format, based on RFC 6020 and RFC 7951, that describes the path to the modeled root, including all list names, list keys, and list key values. This path can be used with the pySROS libraries. For more information, see the *7705 SAR Gen 2 System Management Guide*, "Python" chapter.

```
*[ex:/configure card 1 mda 2 network]
A:admin@node-2# pwc cli-pathA:admin@cses-V93# pwc model-path
Present Working Context:
/nokia-conf:configure/card=1/mda=2/network

*[ex:/configure card 1 mda 2 network]
A:admin@cses-V93# pwc gnmi-path
Present Working Context:
/configure/card[slot-number=1]/mda[mda-slot=2]/network

*[ex:/configure card 1 mda 2 network]
A:admin@cses-V93# pwc cli-path
Present Working Context:
/configure card 1 mda 2 network

*[ex:/configure card 1 mda 2 network]
A:admin@cses-V93# pwc json-instance-path
Present Working Context:
/nokia-conf:configure/card[slot-number="1"]/mda[mda-slot="2"]/network
```

- **back**

  The **back** command can be used to go back one or more levels. If no parameter value is specified for the number of levels to go back, the default is one level. Using **back** at the top of the current command tree moves the context to the operational root level. If the number of levels specified is greater than the current depth, the context moves to the operational root. A closing brace (}) can also be used to go back one level.

```
*[ex:/configure card 1 mda 2 network]
A:admin@node-2# back

*[ex:/configure card 1 mda 2]
A:admin@node-2# back 2

*[ex:/configure]
A:admin@node-2# back 5

*[ex:/]
A:admin@node-2#
```

- **top**

  The **top** command moves the context to the top of the current command tree without exiting the mode. Use the **top** command instead of issuing the **back** command multiple times to move the context to the top of the command tree.

```
*(ex)[/]
A:admin@node-2# configure

*(ex)[/configure]
A:admin@node-2# card 1

*(ex)[/configure card 1]
A:admin@node-2# mda 2

*(ex)[/configure card 1 mda 2]
A:admin@node-2# network

*(ex)[/configure card 1 mda 2 network]
A:admin@node-2# top

*(ex)[/configure]
A:admin@node-2#
```

- **exit**

  The **exit** command moves the context to the previous context in the current command tree. If the previous context was up one level, the **exit** command functions similarly to the **back** command. Using **exit all** moves the context to the operational root. A slash (/) or **Ctrl-Z** can also be used instead of **exit all**. Using **exit** at the operational root has no effect. To log out of the system, the **logout** command must be used.

```
*(ex)[/]
A:admin@node-2#

*(ex)[/]
A:admin@node-2# configure card 1 mda 2

*(ex)[/configure card 1 mda 2]
A:admin@node-2# network
```

```
*(ex)[/configure card 1 mda 2 network]
A:admin@node-2# exit all

*(ex)[/]
A:admin@node-2# configure card 1 mda 2 network

*(ex)[/configure card 1 mda 2 network]
A:admin@node-2# /

*(ex)[/]
A:admin@node-2#
```

2.7 Using the tree command

The **tree** command displays the command tree under the present working context, excluding the present working context element. Hierarchy is indicated with a pipe (|), and a "**+--** " separator precedes each element. The tree output is in alphabetical order of elements.

Example

```
[ex:/configure system security aaa remote-servers]
A:admin@node-2# tree
+-- apply-groups
+-- apply-groups-exclude
+-- ldap
|   +-- admin-state
|   +-- apply-groups
|   +-- apply-groups-exclude
|   +-- public-key-authentication
|   +-- route-preference
|   +-- server
|   |   +-- address
|   |   |   +-- apply-groups
|   |   |   +-- apply-groups-exclude
|   |   |   +-- port
|   |   +-- admin-state
|   |   +-- apply-groups
|   |   +-- apply-groups-exclude
|   |   +-- bind-authentication
|   |   |   +-- password
|   |   |   +-- root-dn
|   |   +-- search
|   |   |   +-- base-dn
|   |   +-- server-name
|   |   +-- tls-profile
|   +-- server-retry
|   +-- server-timeout
|   +-- use-default-template

---snip---
```

2.7.1 Using the flat option

The **flat** option displays the command hierarchy under the present working context on one line, excluding the present working context element.

Example

```
[/]
A:admin@node-2# tree flat
admin
admin clear
admin clear security
admin clear security lockout
admin clear security lockout all
admin clear security lockout user
admin clear security password-history
admin clear security password-history all
admin clear security password-history user
admin disconnect
admin disconnect address
admin disconnect op-table-bypass
admin disconnect session-id
admin disconnect session-type
admin disconnect username
admin reboot
admin reboot [card]
admin reboot now
admin redundancy
admin redundancy force-switchover
admin redundancy force-switchover now
admin redundancy synchronize
admin redundancy synchronize boot-environment
admin redundancy synchronize certificate
admin redundancy synchronize configuration

---snip---
```

2.7.2 Using the detail option

The **detail** option displays all key and field values in the output on every line.

Example

```
[/]
A:admin@node-2# tree detail
+-- admin
| +-- clear
| | +-- security
| | +-- lockout
| | | +-- all
| | | +-- user <string>
| | +-- password-history
| | +-- all
| | +-- user <string>
| +-- disconnect
| | +-- address <ipv4-address | ipv6-address>
| | +-- op-table-bypass <boolean>
| | +-- session-id <number>
| | +-- session-type <keyword>
| | +-- username <string>
| +-- reboot
| | +-- [card] <keyword>
| | +-- now
| +-- redundancy
| | +-- force-switchover
```

```
| | | +-- now
| | +-- synchronize
| | +-- boot-environment
| | +-- certificate
| | +-- configuration

---snip---
```

The **flat** and **detail** options can be combined in any order.

Example

```
[/]
A:admin@node-2# tree flat detail
admin
admin clear
admin clear security
admin clear security lockout
admin clear security lockout all
admin clear security lockout user <string>
admin clear security password-history
admin clear security password-history all
admin clear security password-history user <string>
admin disconnect
admin disconnect address <ipv4-address | ipv6-address>
admin disconnect op-table-bypass <boolean>
admin disconnect session-id <number>
admin disconnect session-type <keyword>
admin disconnect username <string>
admin reboot
admin reboot [card] <keyword>
admin reboot now
admin redundancy
admin redundancy force-switchover
admin redundancy force-switchover now
admin redundancy synchronize
admin redundancy synchronize boot-environment
admin redundancy synchronize certificate
admin redundancy synchronize configuration

---snip---
```

2.7.3 Using the path-format option

The **path-format** option is an unnamed command option. This means it can be included with one of its values or omitted in favor of one of its values, which are as follows: **model-path**, **gnmi-path**, or **json-instance-path**.

Note: The **detail** and **flat** tree command options are not available when using the **path-format** option.

The **path-format** option displays in alternative formats the MD-CLI items that are available on the router. The alternative formats may be used in a variety of applications and tools, including for gNMI streaming telemetry and Python programming.

The command outputs all available paths on the router. Not all the paths are available to be used in external applications but all are provided for reference.

Example

```
[ex:/configure]
A:admin@node-2# tree model-path
/nokia-conf:configure/aaa
/nokia-conf:configure/aaa/apply-groups
/nokia-conf:configure/aaa/apply-groups-exclude
/nokia-conf:configure/aaa/diameter
/nokia-conf:configure/aaa/diameter/node
/nokia-conf:configure/aaa/diameter/node/apply-groups
/nokia-conf:configure/aaa/diameter/node/apply-groups-exclude
/nokia-conf:configure/aaa/diameter/node/connection
/nokia-conf:configure/aaa/diameter/node/connection/ipv4
/nokia-conf:configure/aaa/diameter/node/connection/ipv4/allow-connections
/nokia-conf:configure/aaa/diameter/node/connection/ipv4/local-address
/nokia-conf:configure/aaa/diameter/node/connection/ipv6
/nokia-conf:configure/aaa/diameter/node/connection/ipv6/allow-connections
/nokia-conf:configure/aaa/diameter/node/connection/ipv6/local-address
/nokia-conf:configure/aaa/diameter/node/connection/timer
/nokia-conf:configure/aaa/diameter/node/description
/nokia-conf:configure/aaa/diameter/node/origin-realm
```

Example

```
[ex:/configure]
A:admin@node-2# tree gnmi-path
/configure/aaa
/configure/aaa/apply-groups
/configure/aaa/apply-groups-exclude
/configure/aaa/diameter
/configure/aaa/diameter/node[origin-host]
/configure/aaa/diameter/node[origin-host]/apply-groups
/configure/aaa/diameter/node[origin-host]/apply-groups-exclude
/configure/aaa/diameter/node[origin-host]/connection
/configure/aaa/diameter/node[origin-host]/connection/ipv4
/configure/aaa/diameter/node[origin-host]/connection/ipv4/allow-connections
/configure/aaa/diameter/node[origin-host]/connection/ipv4/local-address
/configure/aaa/diameter/node[origin-host]/connection/ipv6
/configure/aaa/diameter/node[origin-host]/connection/ipv6/allow-connections
/configure/aaa/diameter/node[origin-host]/connection/ipv6/local-address
/configure/aaa/diameter/node[origin-host]/connection/timer
/configure/aaa/diameter/node[origin-host]/description
/configure/aaa/diameter/node[origin-host]/origin-realm
```

Example

```
[ex:/configure]
A:admin@node-2# tree json-instance-path
/nokia-conf:configure/aaa
/nokia-conf:configure/aaa/apply-groups
/nokia-conf:configure/aaa/apply-groups-exclude
/nokia-conf:configure/aaa/diameter
/nokia-conf:configure/aaa/diameter/node[origin-host]
/nokia-conf:configure/aaa/diameter/node[origin-host]/apply-groups
/nokia-conf:configure/aaa/diameter/node[origin-host]/apply-groups-exclude
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv4
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv4/allow-connections
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv4/local-address
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv6
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv6/allow-connections
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/ipv6/local-address
```

```
/nokia-conf:configure/aaa/diameter/node[origin-host]/connection/timer
/nokia-conf:configure/aaa/diameter/node[origin-host]/description
/nokia-conf:configure/aaa/diameter/node[origin-host]/origin-realm
```

2.8 Using control characters and editing keystrokes on the command line

Table 3: Control characters and keystrokes available to execute and edit commands

Command	Description
/ (Slash)	Return to the operational root (equivalent to **exit all**) if used without parameters. Navigate into context or set the value and remain in current context if used at the beginning of a line (equivalent to **exit all**, and then the command)
} (Closing brace)	Go back one level
Alt-B or **Esc+B**	Move back one word or to the beginning of the current word if the cursor is not at the start of the word
Alt-D or **Esc+D**	Delete the remainder of the word
Alt-F or **Esc+F**	Move forward one word
Ctrl-Z	Return to operational root. If using **Ctrl-Z** after a command, return to the operational root after executing the command (equivalent to pressing **Enter** after the command and **exit all** after the command has executed).
Ctrl-C	Stop the current command
Ctrl-D	Delete the current character
Ctrl-W	Delete the word up to the cursor
Ctrl-H	Delete the current character and move the cursor left
Ctrl-U	Delete text up to the cursor and preserve the character under the cursor
Ctrl-K	Delete the text after the cursor, without preserving the character under the cursor
Ctrl-A or Home	Move to the beginning of the line
Ctrl-E or End	Move to the end of the line
Ctrl-P or Up arrow	Display the previous command from the command history

Command	Description
Ctrl-N or Down arrow	Display the next command from the command history
Ctrl-R	Search the command history in reverse order
Esc+.	Display the last element of the previous command
Ctrl-B or Left arrow	Move the cursor one space to the left
Ctrl-F or Right arrow	Move the cursor one space to the right
Ctrl-I	Enter a tab
Ctrl-J	Enter a new line
Ctrl-M	Enter a carriage return
Ctrl-L	Clear the screen

2.9 Displaying available commands using Tab

Variables, keywords, global commands, and configuration commands and units are separated by a blank line in the output, in the following order:

1. values or units (mutually exclusive)
2. keywords
3. global commands
4. configuration commands

Example: Displaying commands using Tab

```
[ex:/configure log]
A:admin@node-2# Press Tab

 accounting-policy        app-route-notifications  apply-groups
 apply-groups-exclude     event-damping            event-handling
 event-trigger            file                     filter
 foo                      log-events               log-id
 route-preference         services-all-events      snmp-trap-group
 syslog                   throttle-rate

 back                     delete                   edit-config
 enable                   exec                     exit
 history                  info                     insert
 logout                   oam                      ping
 pwc                      pyexec                   ssh
 telnet                   top                      traceroute
 tree

 annotate                 commit                   compare
```

```
 copy                    discard                 rename
 update                  validate
```

```
[ex:/configure log]
A:admin@node-2# event-damping Press Tab

 <event-damping>
 false
 true

 accounting-policy       app-route-notifications  apply-groups
 apply-groups-exclude    event-handling           event-trigger
 file                    filter                   log-events
 log-id                  route-preference         services-all-events
 snmp-trap-group         syslog                   throttle-rate

 delete                  insert
```

The ? help displays similar information but only displays global or configuration commands at the operational root or at the root of a command context.

Example: Global commands displayed at the environment root context

```
[/environment]
A:admin@node-2# ?

 command-completion    + Enter the command-completion context
 console               + Enter the console context
 message-severity-     + Enter the message-severity-level context
  level
 more                  - Prompt to continue or stop when output text fills page
 progress-indicator    + Enter the progress-indicator context
 prompt                + Enter the prompt context
 time-display          - Time zone displayed before the prompt
 time-format           - Time format to display date and time

 Global commands:
 back                  - Move back one or more levels
 delete                - Delete an element
 edit-config           - Enter a candidate configuration mode
 enable                - Enable administrative mode
 exec                  - Execute commands from a file
 exit                  - Return to the previous context or to operational root
 history               - Show the command history
 info                  - Show the information under the present working context
 logout                - Exit the CLI session
 oam                   - Perform OAM tests
 ping                  - Ping an IP address or DNS name
 pwc                   - Show the present working context
 pyexec                - Execute a Python application
 ssh                   - SSH to an IP address or DNS name
 telnet                - Telnet to an IP address or DNS name
 top                   - Move to the top level of the context
 traceroute            - Show the route taken to an IP address or DNS name
 tree                  - Show the command tree under the present working context
```

Example: Global commands available in the context

The global commands are not displayed in the environment prompt context, for example, although these commands are still available in the context.

```
[/environment prompt]
A:admin@node-2# ?

 context               - Show the current command context in the prompt
 newline               - Add a new line before every prompt line
 timestamp             - Show the timestamp before the first prompt line
 uncommitted-changes-  - Show an asterisk (*) when uncommitted changes exist
  indicator

[/environment prompt]
A:admin@node-2# info detail
    context true
    newline true
    timestamp false
    uncommitted-changes-indicator true
```

2.9.1 Available commands with mutually exclusive commands

When a command that is part of a choice of commands is entered at the MD-CLI command prompt, the other mutually exclusive commands are no longer available to be entered on the same prompt line. Other commands that are not associated with the particular choice commands are still available.

Example

In the following example, if either the **cir** or **fir** command is entered, the **police** command is not available. The **pir** command is available regardless of which choice command is entered.

```
*[ex:/configure qos sap-ingress "ing-1" queue 1 rate]
A:admin@node-2# ?

 pir                   - Administrative PIR

 Choice: rate-cir-fir-or-police
 cir                  :- Administrative CIR
 fir                   - Administrative FIR
 police               :- Drop the traffic feeding into queue above the PIR rate

*[ex:/configure qos sap-ingress "ing-1" queue 1 rate]
A:admin@node-2# cir max Press Tab

 fir     pir

 delete
```

Example

Similarly, if the **police** command is entered, the **cir** and **fir** commands are unavailable on the same command prompt line.

```
*[ex:/configure qos sap-ingress "ing-1" queue 1 rate]
A:admin@node-2# police ?

 police - This element has no values
```

```
    Drop the traffic feeding into queue above the PIR rate

 pir                   - Administrative PIR

*[ex:/configure qos sap-ingress "ing-1" queue 1 rate]
A:admin@node-2# police Press Tab

 pir

 delete
```

2.10 Using command completion

The MD-CLI supports both command abbreviation and command completion. When typing a command, **Tab**, **Spacebar**, or **Enter** invokes auto-completion. If the text entered is enough to match a specific command, auto-completion completes the command. If the text entered is not sufficient to identify a specific command, pressing **Tab** or **Spacebar** displays options in alphabetical order matching the text entered.

The **environment command-completion** command controls what keystrokes can trigger command completion. Each keystroke is independently controlled with its own Boolean value.

Example

```
(ex)[/environment command-completion]
A:admin@node-2# info detail
    enter true
    space true
    tab true
```

Note: If **Spacebar** completion has multiple matches and also matches an keyword, the space is considered a separator and auto-completion is not triggered.

- **configure por**+**Spacebar** displays auto-completion results
- **configure port**+**Spacebar** inserts a space and suppresses auto-completion results
- **configure por**+**Tab** displays auto-completion results
- **configure port**+**Tab** displays auto-completion results

2.10.1 Variable parameter completion

Variable parameter completion works only with the **Tab** key. All configured variables from the candidate and running configuration datastores are displayed. Line wrapping may occur for variables with long names. Parameters are displayed in alphabetical or numerical order. The variable parameter name is always displayed as the first line.

Example

In the following example, "interface-name" is the variable parameter name and "int-1" and "system" are configured names.

```
*[ex:/configure router "Base"]
A:admin@node-2# interface Press Tab
```

```
 <interface-name>
 "int-1"
 "system"

*[ex:/configure router "Base"]
A:admin@node-2# interface
```

2.10.1.1 Completion for lists with a default keyword

Some list elements have a default keyword defined, such as the **router** command, where the default keyword is "Base". When the command completion parameters (**Enter**, **Spacebar**, and **Tab**) are at their default settings (**true**), and the initial input matches an element in the list and a unique command keyword, the matching keyword is completed instead of the variable.

For example, the **router** command has a default keyword defined as "Base". If router "bf" is created using the command **configure router "bf"** (with quotation marks), and there is an existing **bfd** command context, the variable completion is as follows.

Example: Display for router+Spacebar+Tab

```
*[ex:/configure]
A:admin@node-2# router Press Tab

 <router-name>
 "Base"
 "bf"
 "management"

 aggregates             allow-icmp-redirect    allow-icmp6-redirect
 apply-groups           apply-groups-exclude   autonomous-system
 bfd                    bgp                    bier

---snip---
```

Example: Display for router bf+Tab

```
*[ex:/configure]
A:admin@node-2# router bfPress Tab

 "bf"

 bfd
```

Example: Entering router bf+Enter

Entering **router bf**+**Enter** completes to **router bfd** and enters the **router "Base" bfd** context:

```
*[ex:/configure]
A:admin@node-2# router bfd Press Enter

*[ex:/configure router "Base" bfd]
A:admin@node-2#
```

Example: Entering router bf+Spacebar

Similarly, **router bf**+**Spacebar** completes to **router bfd** and enters the **router "Base" bfd** context when **Enter** is pressed:

```
*[ex:/configure]
A:admin@node-2# router bfd Press Spacebar+Enter

*[ex:/configure router "Base" bfd]
A:admin@node-2#
```

Example: Using quotation marks to specify a variable

To enter the context for router "bf", use quotation marks to specify the variable:

```
[ex:/configure]
A:admin@node-2# router "bf"

*[ex:/configure router "bf"]
A:admin@node-2#
```

Example: Command completion for enter set to false

If the command completion for **enter** is set to **false**, **router bf**+ **Enter** allows the match to router "bf". Similarly, when the command completion for **space** is **false**, **router bf**+**Spacebar** also matches to router "bf" instead of the **bfd** context

```
*(ex)[/environment command-completion]
A:admin@node-2# info detail
    enter true
    space true
    tab true

*(ex)[/environment command-completion]
A:admin@node-2# enter false

*(ex)[/environment command-completion]
A:admin@node-2# space false

*(ex)[/environment command-completion]
A:admin@node-2#

*(ex)[/]
A:admin@node-2# configure

*(ex)[/configure]
A:admin@node-2# router bf Press Enter

*(ex)[/configure router "bf"]
A:admin@node-2# back

*(ex)[/configure]
A:admin@node-2# router bf Press Spacebar+Enter

*(ex)[/configure router "bf"]
A:admin@node-2#
```

2.10.1.2 Completion for keyword-based leaf-lists

For keyword-based leaf-lists, command completion displays all possible values, not only those that are configured. When deleting values in a leaf-list, only the values that are currently configured are displayed.

Example

In the following example, when defining the forwarding traffic classes, all keyword values are listed. When deleting the forwarding traffic classes, only the configured classes are displayed.

```
*[ex:/configure policy-options policy-statement "ss" entry 3 from]
A:admin@node-2# family ?

 family <value>
 family [<value>...] - 1..20 system-ordered values separated by spaces enclosed
                       by brackets

 <value>   - <keyword>
 <keyword> - (ipv4|vpn-ipv4|ipv6|mcast-ipv4|vpn-ipv6|l2-vpn|mvpn-ipv4|mdt-
             safi|ms-pw|flow-ipv4|route-target|mcast-vpn-ipv4|mvpn-ipv6|
             flow-ipv6|evpn|mcast-ipv6|label-ipv4|label-ipv6|bgp-ls|mcast-
             vpn-ipv6|sr-policy-ipv4|sr-policy-ipv6|flow-vpn-ipv4|flow-
             vpn-ipv6)

    Address family as the match condition
```

```
*[ex:/configure policy-options policy-statement "ss" entry 3 from]
A:admin@node-2# family [ipv4 mcast-ipv4 mcast-vpn-ipv4 label-ipv4]

*[ex:/configure policy-options policy-statement "ss" entry 3 from]
A:admin@node-2# info
    family [ipv4 mcast-ipv4 mcast-vpn-ipv4 label-ipv4]

*[ex:/configure policy-options policy-statement "ss" entry 3 from]
A:admin@node-2# delete family Press Tab

 <family>
 ipv4
 mcast-ipv4
 mcast-vpn-ipv4
 label-ipv4
 *
```

2.10.1.3 Completion for Boolean elements

The explicit use of the keyword **true** for a Boolean element is optional. If neither **true** or **false** is entered, the keyword **true** is assumed.

Example

```
(ex)[/environment]
A:admin@node-2# more ?

 more <boolean>
 <boolean> - ([true]|false)
 Default   - true
```

```
    Activate the pager when output is longer than a screen
```

When **Tab** is used for command completion with Boolean elements, the values of **false** and **true** are displayed, along with the names of possible elements that can follow.

Example

In the following example of the **environment more** command, the commands **command-completion**, **console**, **message-severity-level**, and so on, can be defined following the **more** command.

```
(ex)[/environment]
A:admin@node-2# more Press Tab

 <more>
 false
 true

 command-completion      console                 message-severity-level
 progress-indicator      prompt                  time-display
 time-format

 delete
```

2.11 Modifying the idle timeout value for CLI sessions

A single idle timeout applies to all CLI engines in a CLI session (classic and MD-CLI). The idle timeout can be modified to a value between 1 and 1440 minutes.

The following points apply:

- The idle timeout only affects new CLI sessions. Existing and current sessions retain the previous idle timeout.
- The idle timeout can be disabled by setting the value to **none**.
- The "Idle time" column in the **show users** display is reset after an action in either CLI engine.

Example

A warning message is displayed when a session reaches one-half the value of the idle timeout, and another message is displayed when the idle timeout expires and the user is logged out.

```
[/]
A:admin@node-2# show users
================================================================================
User                            Type      Login time           Idle time
  Session ID   (Router instance)
               From
================================================================================
                                Console         --             0d 21:44:02 --
  6            (--)
               --
admin                           SSHv2     16FEB2021 20:42:15   0d 00:00:04 --
  12           (management)
               192.168.144.97
admin                           SSHv2     16FEB2021 19:49:45   0d 00:00:00 --
 #11           (management)
               192.168.144.97
```

```
-------------------------------------------------------------------------------
Number of users: 2
'#' indicates the current active session
===============================================================================
```

2.11.1 Idle timeout interaction with the classic CLI

The idle timeout configured in the classic CLI affects all new sessions as well as the current session. However, the current session is only affected if the classic CLI engine is active when the idle timeout expires. Configuration changes via the MD-CLI or any other interface only affect new sessions that begin after the change.

2.12 Using the pager

The MD-CLI pager paginates command output one screenful at a time. It supports movement and searching in forward and backward directions.

The status line at the bottom of the screen displays the following information.

Example

```
--(more)--(p%)--(lines b-e/t)--
```

where

- p - indicates the position percentage in the output
- b - indicates the beginning line of the output on the screen
- e - indicates the ending line of the output on the screen
- t - indicates the total lines in the output

```
--(more)--(5%)--(lines 1-29/527)—
```

If the command output is still being generated, a question mark ("?") is displayed after the total lines and a warning indicates that the search is incomplete. These indicators are removed when the command output completes.

```
--(more)--(0%)--(lines 1-29/49833?)--(warning: search will be incomplete)--
```

The pager supports a high number of output lines. If the command output exceeds the pager memory, an asterisk (*) is displayed after the total lines and a warning indicates that the search is incomplete.

```
--(more)--(0%)--(lines 3578-3607/93151747*)--(warning: search will be incomplete)--
```

The following table describes the general and movement commands available for use.

Note: (n) indicates that a command may be preceded by a number *n*.

Table 4: General and movement commands

Action	Command
Show the help	**h**, **H**
Exit the pager	**q**, **Q**, **Ctrl-C**, **Ctrl-Z**
Move a line forward (n)	**e**, **j**, **Ctrl-E**, **Ctrl-J**, **Ctrl-M**, **Ctrl-N**, Down arrow, Enter
Move a half screen forward (n)	**d**, **Ctrl-D**, **Ctrl-I**, Tab
Move a screen forward (n)	**f**, **Ctrl-F**, **Ctrl-V**, Page down, Right arrow, Spacebar
Move a line backward (n)	**k**, **y**, minus (**-**), **Ctrl-8**, **Ctrl-H**, **Ctrl-K**, **Ctrl-P**, **Ctrl-Y**, Backspace, Delete, Up arrow
Move a half screen backward (n)	**u**, **Ctrl-U**
Move a screen backward (n)	**b**, **Ctrl-B**, Left arrow, Page up
Move to the beginning of the output	**g**, **p**, open angle bracket (**<**), **Ctrl-A**, Home
Move to the end of the output	**G**, closed angle bracket (**>**), End
Jump to line (n)	**g**, **G**

You can search the output by entering a search command followed by a string or regular expression at the bottom of the screen. The first matching result in the output is highlighted and underlined, and all visible matches on the screen are underlined. Repeating the search displays the next match, and reentering the search command recalls the previous search, allowing you to edit it.

Example: Searching with a string

The following string search matches any string containing "mda".

```
/mda
```

Example: Searching with a regular expression

The following regular expression search matches the neighbor with the IP address that ends with 42 (**neighbor "10.81.192.42"**).

```
/nei.*42
```

See Using regular expressions for more information about using regular expressions. The regular expression is highlighted until it is valid. Highlighting occurs when delimiters such as () and [] are unmatched, or when an asterisk (*) or question mark (?) is entered without a preceding expression.

The following table describes the search commands available for use.

Table 5: Search commands

Action	Command
Forward search	forward slash (**/**)
Backward search	question mark (**?**)
Repeat previous forward search	**n**
Repeat previous backward search	**N**
Clear search highlighting	**c**, **Ctrl-L**, **Esc+U**

The following usage guidelines apply to the pager:

- the minimum supported terminal size is 80 characters wide by 24 lines long
- resizing the terminal when the pager is active is not supported

2.13 Using output modifiers

Output modifiers provide support for post-processing of CLI output. Output modifiers are specified using a vertical bar (|) character. The following points apply when using output modifiers:

- Output modifiers can be appended to any CLI command in any command context.
- Output modifiers work across soft line breaks (visual lines) that are wrapped because of the terminal width; for example, using **match** or **count**. They do not work across hard line breaks (logical lines).
- Modifiers can be combined in any order. No hard limit exists for the number of combinations. Output is processed linearly and there is little impact on the system performance except to the operator session that entered the modifier combination.

2.13.1 Using | match options

The **| match** options match a pattern in the output.

The following options are supported for use with the **| match** command:

- **ignore-case**

  This specifies to ignore case in pattern match.
- **invert-match**

  This specifies to invert the pattern match selection.
- **max-count**

  This specifies the maximum number of displayed matches.
- **post-lines**

  This specifies the number of lines to display following the matched line.
- **pre-lines**

  This specifies the number of lines to display preceding the matched line.

The default pattern matching is a string match. If the required pattern to match includes a space, the pattern must be delimited by quotation marks ("). Regular expressions are delimited by apostrophes ('). See Using regular expressions for more information about using regular expressions.

Example

The following example matches on the pattern **autonomous-system** in the **tree detail** under the **configure router "Base"** context, and starts the display with ten lines preceding the pattern match.

```
[ex:/configure router "Base"]
A:admin@node-2# tree detail | match autonomous-system pre-lines 10
|   |   +-- local-preference <number>
|   |   +-- policy <reference>
|   |   +-- summary-only <boolean>
|   |   +-- tunnel-group <number>
|   +-- apply-groups <reference>
|   +-- apply-groups-exclude <reference>
+-- allow-icmp-redirect <boolean>
+-- allow-icmp6-redirect <boolean>
+-- apply-groups <reference>
+-- apply-groups-exclude <reference>
+-- autonomous-system <number>

---snip---
```

2.13.2 Using the | count option

The **| count** option displays the line count of the output.

Note: Error messages are not processed by output modifiers. They are always displayed and are not affected by the **count** or **match** modifiers.

Example

```
[ex:/configure router "Base"]
A:admin@node-2# tree flat detail | match seamless-bfd
bfd seamless-bfd
bfd seamless-bfd peer <unicast-ipv4-address | global-unicast-ipv6-address>
bfd seamless-bfd peer <unicast-ipv4-address | global-unicast-ipv6-address> apply-groups
 <reference>
bfd seamless-bfd peer <unicast-ipv4-address | global-unicast-ipv6-address> apply-groups-
exclude <reference>
bfd seamless-bfd peer <unicast-ipv4-address | global-unicast-ipv6-address> discriminator
 <number>

[ex:/configure router "Base"]
A:admin@node-2# tree flat detail | match seamless-bfd | count
Count: 5 lines
```

2.13.3 Using the | no-more option

The **| no-more** option displays the output with pagination disabled for the command. This option is similar to the **environment more false** setting that applies to all commands, where the entire output text is printed without page interruptions.

2.13.4 Using the | reverse-dns option

The **| reverse-dns** option performs a reverse DNS lookup on any IPv4 or IPv6 address in the input to the output modifier. The result of the lookup is inserted as the next line in the output on each line where an IP address is identified. If no match is found, no additional output is printed.

Example

```
[/]
A:admin@node-2# ping 10.184.216.34 | reverse-dns
PING 10.184.216.34 56 data bytes
  (10.184.216.34) www.example.com
64 bytes from 10.184.216.34: icmp_seq=1 ttl=61 time=82.4ms.
64 bytes from 10.184.216.34: icmp_seq=2 ttl=61 time=82.5ms.
64 bytes from 10.184.216.34: icmp_seq=3 ttl=61 time=82.4ms.
64 bytes from 10.184.216.34: icmp_seq=4 ttl=61 time=82.3ms.
64 bytes from 10.184.216.34: icmp_seq=5 ttl=61 time=82.2ms.

---- 10.184.216.34 PING Statistics ----
  (10.184.216.34) www.example.com
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min = 82.2ms, avg = 82.4ms, max = 82.5ms, stddev = 0.122ms
```

2.13.5 Using the | repeat options

The **| repeat** option repeats the command that was entered at a default interval of one second. The option can be configured to run a specified number of times or to display a timestamp.

The following options are supported for use with **| repeat**:

- **[interval]**

  This specifies the repetition interval.

- **count**

  This specifies the repetition count.

- **timestamp**

  This specifies to display the timestamp before each repetition.

Note: The **| count** and **| pyexec** output modifiers execute once after all repetitions. For example, **command | count | repeat** displays the total line count of the output, not the line count after each repetition.

Example: Repeat the command until interrupted

```
[/]
A:admin@node2# show port 1/1/c1/1 statistics | repeat
(output of 1st repetition)
(output of 2nd repetition)
(output of 3rd repetition)
^C
```

Example: Repeat the command at a 10 second interval until interrupted

```
[/]
A:admin@node2# show port 1/1/c1/1 statistics | repeat 10 seconds
(output of 1st repetition)
(output of 2nd repetition)
(output of 3rd repetition)
^C
```

Example: Repeat the command three times

```
[/]
A:admin@node2# show port 1/1/c1/1 statistics | repeat count 3
(output of 1st repetition)
(output of 2nd repetition)
(output of 3rd repetition)
```

Example: Repeat the command three times and display the timestamp

```
[/]
A:admin@node2# show port 1/1/c1/1 statistics | repeat count 3
2024-09-11T20:32:01.22+00:00
(output of 1st repetition)
2024-09-11T20:32:02.22+00:00
(output of 2nd repetition)
2024-09-11T20:32:03.22+00:00
(output of 3rd repetition)
```

2.13.6 Using the file redirect option

Use the **>** option to redirect output to a local or remote file. The **>** must be specified at the end of a command and cannot be combined with other appends or redirects. If the file exists, it is overwritten. If the file does not exist, it is created.

Example

```
[ex:/configure router "Base"]
A:admin@node-2# info detail | match leak-export > ?

 [url] <string>
 <string> - <1..199 characters>

    Location to save the output

[ex:/configure router "Base"]
A:admin@node-2# info detail | match leak-export > cf3:leak-export.cfg
```

2.13.7 Using the file append option

Use the **>>** option to append output to a local file. The **>>** must be specified at the end of a command and cannot be combined with other appends or redirects. If the file exists, the output is appended. If the file does not exist, it is created.

Example

```
[/state router "Base" bgp statistics]
A:admin@node-2# info detail | match bgp-paths >> ?

 [url] <string>
 <string> - <1..199 characters>

    Local location to append the output

[/state router "Base" bgp statistics]
A:admin@node-2# info detail | match bgp-paths >> cf3:bgp-paths
```

2.13.8 Using regular expressions

Regular expressions (REs) used by the MD-CLI engine are delimited by apostrophes ('); for example, '.*'. REs cannot be delimited by quotation marks (''); for example, ".*" because this indicates a string match.

MD-CLI REs are based on a subset of The Open Group Base Specifications Issue 7 and IEEE Std 1003.1-2008, 2016 Edition REs, as defined in chapter 9. MD-CLI REs only support Extended Regular Expression (ERE) notation as defined in section 9.4. Basic Regular Expression (BRE) notation as defined in section 9.3 is not supported.

In ERE notation, a backslash (\) before a special character is treated as a literal character. Backslashes are not supported before ( ) or { }, as they are in BREs to indicate a bracket expression or marked expression.

Table 6: Special characters in extended regular expressions

Special character	Description
.	Matches any single character
*	Matches the preceding expression zero or more times
?	Matches the preceding expression zero or one time
+	Matches the preceding expression one or more times
[ ]	Matches a single character within the brackets
[^]	Matches a single character not within the brackets
^	Matches the starting position
$	Matches the ending position
( )	Defines a marked subexpression
{m,n}	Matches the preceding expression at least *m* and not more than *n* times
{m}	Matches the preceding expression exactly *m* times
{m, }	Matches the preceding expression at least *m* times

Special character	Description
{ ,n}	Matches the preceding expression not more than *n* times
|	Matches either expression preceding or following the |
\	Escapes and treats the following character as a match criterion
-	Separates the start and end of a range

The following examples show the use of a bracket expression as a matching list expression.

Example: Using no match expressions

The first output does not use any match expressions and therefore shows the entire output.

```
[/]
A:admin@node-2# show port

===============================================================================
Ports on Slot 1
===============================================================================
Port          Admin Link Port    Cfg  Oper LAG/ Port Port Port   C/QS/S/XFP/
Id            State      State   MTU  MTU  Bndl Mode Encp Type   MDIMDX
-------------------------------------------------------------------------------
1/1/1         Down  No   Ghost   8704 8704    - netw null xcme
1/1/2         Up    No   Ghost   1514 1514    - accs null xcme
1/1/3         Up    No   Ghost   1514 1514    - accs null xcme
1/1/4         Up    No   Ghost   1514 1514    - accs null xcme
1/1/5         Up    No   Ghost   1514 1514    - accs null xcme
1/1/6         Down  No   Ghost   8704 8704    - netw null xcme
1/1/7         Down  No   Ghost   8704 8704    - netw null xcme
1/1/8         Down  No   Ghost   8704 8704    - netw null xcme
1/1/9         Down  No   Ghost   8704 8704    - netw null xcme
1/1/10        Down  No   Ghost   8704 8704    - netw null xcme
1/1/11        Down  No   Ghost   8704 8704    - netw null xcme
1/1/12        Down  No   Ghost   8704 8704    - netw null xcme
1/2/1         Up    No   Ghost   8704 8704    - netw null xcme
1/2/2         Up    No   Ghost   1514 1514    - accs null xcme
1/2/3         Up    No   Ghost   1514 1514    - accs null xcme
1/2/4         Down  No   Ghost   8704 8704    - netw null xcme
(more)--(64%)--(lines 1-23/37)--
```

Example: Using matching list expression

In this matching list expression, a match is any single character in the bracket expression, which in this case is 1, 3, or 5.

```
[/]
A:admin@node-2# show port | match '1/1/[135]'
1/1/1         Down  No   Ghost   8704 8704    - netw null xcme
1/1/3         Up    No   Ghost   1514 1514    - accs null xcme
1/1/5         Up    No   Ghost   1514 1514    - accs null xcme
1/1/10        Down  No   Ghost   8704 8704    - netw null xcme
1/1/11        Down  No   Ghost   8704 8704    - netw null xcme
1/1/12        Down  No   Ghost   8704 8704    - netw null xcme
```

Example: Using non-matching list expression

In this non-matching list expression, a match is any single character not in the bracket expression, that is, not 1, 2, or 4.

```
[/]
A:admin@node-2# show port | match '1/1/[^124]'
1/1/3         Up    No   Ghost   1514 1514    - accs null xcme
1/1/5         Up    No   Ghost   1514 1514    - accs null xcme
1/1/6         Down  No   Ghost   8704 8704    - netw null xcme
1/1/7         Down  No   Ghost   8704 8704    - netw null xcme
1/1/8         Down  No   Ghost   8704 8704    - netw null xcme
1/1/9         Down  No   Ghost   8704 8704    - netw null xcme
```

Example: Using the range operator

The range operator (-) can be used in a matching or non-matching list expression.

```
[/]
A:admin@node-2# show port | match '1/1/[3-7]'
1/1/3         Up    No   Ghost   1514 1514    - accs null xcme
1/1/4         Up    No   Ghost   1514 1514    - accs null xcme
1/1/5         Up    No   Ghost   1514 1514    - accs null xcme
1/1/6         Down  No   Ghost   8704 8704    - netw null xcme
1/1/7         Down  No   Ghost   8704 8704    - netw null xcme
```

```
[/]
A:admin@node-2# show port | match '1/1/[^3-7]'
1/1/1         Down  No   Ghost   8704 8704    - netw null xcme
1/1/2         Up    No   Ghost   1514 1514    - accs null xcme
1/1/8         Down  No   Ghost   8704 8704    - netw null xcme
1/1/9         Down  No   Ghost   8704 8704    - netw null xcme
1/1/10        Down  No   Ghost   8704 8704    - netw null xcme
1/1/11        Down  No   Ghost   8704 8704    - netw null xcme
1/1/12        Down  No   Ghost   8704 8704    - netw null xcme
```

Example: Using the alternation operator

The alternation operator (|) can be used with or without a bracket expression to match against two or more alternative expressions.

```
[/]
A:admin@node-2# show port | match '1/1/[2-5|7-9]'
1/1/2         Up    No   Ghost   1514 1514    - accs null xcme
1/1/3         Up    No   Ghost   1514 1514    - accs null xcme
1/1/4         Up    No   Ghost   1514 1514    - accs null xcme
1/1/5         Up    No   Ghost   1514 1514    - accs null xcme
1/1/7         Down  No   Ghost   8704 8704    - netw null xcme
1/1/8         Down  No   Ghost   8704 8704    - netw null xcme
1/1/9         Down  No   Ghost   8704 8704    - netw null xcme
```

Example: Using no bracket expression

Without a bracket expression, an exact match is attempted against two or more alternative expressions.

```
*[ex:/configure card 1]
A:admin@node-2# info | match '10g|100g'
        mda-type imm4-10gb-xp-xfp
```

```
        mda-type cx2-100g-cfp
```

MD-CLI REs match on the output format of an element, as is shown in the configuration. For example, if the value of an element is shown in hexadecimal in **info** output, a decimal RE does not match the value.

Example: Using anchoring special characters

MD-CLI REs are not implicitly anchored. This complete configuration is used to show how the ^ or $ anchoring special characters can be used to match in the following examples.

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info
    group "external" {
    }
    group "internal" {
    }
    neighbor "192.168.10.1" {
        group "external"
        keepalive 30
        peer-as 100
    }
    neighbor "192.168.10.2" {
        group "external"
        peer-as 100
        family {
            ipv4 true
        }
    }
```

Example: Using the anchor character preceded by four spaces

This example uses the ^ anchor character to match on "group" preceded by four spaces at the beginning of the line.

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info | match '^    group' pre-lines 1
    group "external" {
    }
    group "internal" {
```

Example: Using the anchor character preceded by eight spaces

This example uses the ^ anchor character to match on "group" preceded by eight spaces at the beginning of the line.

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info | match '^        group' pre-lines 1
    neighbor "192.168.10.1" {
        group "external"
    neighbor ""192.168.10.2" {
        group "external"
```

Example: Using the compare command

In the following configuration example using the **compare** command, the **| match** option filters out those commands to be deleted (configuration statements beginning with minus (-)) and those to be added (configuration statements beginning with plus (+)).

```
*[ex:/configure log accounting-policy 5]
A:admin@node-2# /compare
+   admin-state enable
-   collection-interval 105
+   collection-interval 75
-   include-system-info true
+   include-system-info false

*[ex:/configure log accounting-policy 5]
A:admin@node-2# /compare | match '^-'
-   collection-interval 105
-   include-system-info true
```

Example: Using the backslash

The backslash (\) is used to match the literal "**+**" character that denotes additions to the configuration seen in the **compare** command.

```
*[ex:/configure log accounting-policy 5]
A:admin@node-2# /compare | match '^\+'
+   admin-state enable
+   collection-interval 75
+   include-system-info false
```

A character class expression is expressed as a character class name enclosed within bracket colon ("[:" and ":]") delimiters.

Character class expressions must be enclosed within brackets. The expression '[[:digit:]]' is treated as an RE containing the character class "digit", while '[:digit:]' is treated as an RE matching ":", "d", "i", "g", or "t".

Collating symbols and equivalence classes are not supported in MD-CLI REs.

Table 7: Character class expressions

Character class	Characters matched[1]	Description
[:alnum:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789'	Alphanumeric characters
[:alpha:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz'	Alphabetic characters
[:blank:]	' \t'	Spacebar and Tab
[:cntrl:]	'\007\b\t\n\v\f\r\1\2\3\4\5\6\16\17\20 \21\22\ 23\24\25\26\27\30 \31\32\33\34\35\36\37\ 177'	Control characters

[1] Characters matching the character class are delimited by apostrophes (')

Character class	Characters matched[1]	Description
[:digit:]	'0123456789'	Digits
[:graph:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789 !\"#$%&'()*+,-./:;<=>?@[\\]^_ `{|}~'	Visible characters
[:lower:]	'abcdefghijklmnopqrstuvwxyz'	Lowercase letters
[:print:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789! \"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~ '	Visible characters and the space character
[:punct:]	'!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~'	Punctuation characters
[:space:]	'\t\n\v\f\r '	Whitespace (blank) characters
[:upper:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ'	Uppercase letters
[:xdigit:]	'0123456789ABCDEFabcdef'	Hexadecimal digits

2.14 Navigating contexts

2.14.1 Entering contexts

Configuring a container navigates into the context.

Example: First container is log and next is log-events

All containers are marked with a "+".

```
[ex:/configure log]
A:admin@node-2# ?

 accounting-policy     + Enter the accounting-policy list instance
 app-route-            + Enter the app-route-notifications context
  notifications
 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 event-damping         - Allow event damping algorithm to suppress QoS or
                         filter change events
 event-handling        + Enter the event-handling context
 event-trigger         + Enter the event-trigger context
 file                  + Enter the file list instance
 filter                + Enter the filter list instance
 log-events            + Enter the log-events context
 log-id                + Enter the log-id list instance
 route-preference      + Enter the route-preference context
 services-all-events   + Enter the services-all-events context
```

[1] Characters matching the character class are delimited by apostrophes (')

```
 snmp-trap-group       + Enter the snmp-trap-group list instance
 syslog                + Enter the syslog list instance
 throttle-rate         + Enter the throttle-rate context

[ex:/configure log]
A:admin@node-2# log-events

[ex:/configure log log-events]
A:admin@node-2#
```

Example: Same context entered on one line

Alternatively, the same context can be entered on one line:

```
(ex)[/]
A:admin@node-2# configure log log-events

(ex)[/configure log log-events]
A:admin@node-2#
```

Example: Container lists are marked with a "+"

Container lists are also marked with a "+" and the context is entered by specifying the key for the list.

```
[ex:/configure log]
A:admin@node-2# ?

 accounting-policy     + Enter the accounting-policy list instance
 app-route-            + Enter the app-route-notifications context
  notifications
 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 event-damping         - Allow event damping algorithm to suppress QoS or
                         filter change events
 event-handling        + Enter the event-handling context
 event-trigger         + Enter the event-trigger context
 file                  + Enter the file list instance
 filter                + Enter the filter list instance
 log-events            + Enter the log-events context
 log-id                + Enter the log-id list instance
 route-preference      + Enter the route-preference context
 services-all-events   + Enter the services-all-events context
 snmp-trap-group       + Enter the snmp-trap-group list instance
 syslog                + Enter the syslog list instance
 throttle-rate         + Enter the throttle-rate context
```

```
[ex:/configure log]
A:admin@node-2# log-id ?

 [name] <string>
 <string> - <1..64 characters>

    Log ID
```

```
[ex:/configure log]
A:admin@node-2# log-id "99"

[ex:/configure log log-id "99"]
A:admin@node-2#
```

Example: Configuring a leaf element

Configuring a leaf element maintains the present working context if there is no explicit opening brace. Entering an explicit opening brace navigates into the specified context.

```
*[ex:/configure card 1 mda 2]
A:admin@node-2# clock-mode mode ?

 mode <keyword>
 <keyword> - (adaptive|differential)

    Clock mode

*[ex:/configure card 1 mda 2]
A:admin@node-2# clock-mode mode adaptive

*[ex:/configure card 1 mda 2]
A:admin@node-2# clock-mode { mode adaptive

*[ex:/configure card 1 mda 2 clock-mode]
A:admin@node-2#
```

Example: Configuring a container

Configuring a container navigates into the context.

```
[ex:/configure router "Base"]
A:admin@node-2# ?

 aggregates            + Enter the aggregates context
 allow-icmp-redirect   - Allow ICMP redirects on the management interface
 allow-icmp6-redirect  - Allow IPv6 ICMP redirects on the management interface
 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 autonomous-system     - AS number advertised to peers for this router
 bfd                   + Enter the bfd context
 bgp                   + Enable the bgp context

---snip---

[ex:/configure router "Base"]
A:admin@node-2# bgp

[ex:/configure router "Base" bgp]
A:admin@node-2# ?

 add-paths             + Enable the add-paths context
 admin-state           - Administrative state of the BGP instance
 advertise-external    + Enter the advertise-external context
 advertise-inactive    - Advertise inactive BGP routes to peers

---snip---

[ex:/configure router "Base" bgp]
A:admin@node-2# add-paths

*[ex:/configure router "Base" bgp add-paths]
A:admin@node-2#
```

Configuring an empty container or a list where the only children are keys does not navigate into the context. These elements are displayed with aggregated braces with a space ({ }) on the same line.

It is possible to enter the element name with an opening brace; however, no options are available in this context.

Example: Configuring the list element sdp-include

For example, configuring the list element **sdp-include** with a key of "ref_group_name" does not change the existing context.

```
*[ex:/configure service pw-template "tt"]
A:admin@node-2# sdp-include ref_group_name

*[ex:/configure service pw-template "tt"]
A:admin@node-2# info
    sdp-include "ref_group_name" { }

*[ex:/configure service pw-template "tt"]
A:admin@node-2#
```

2.14.2 Exiting contexts

The **back** and **top** commands are used to navigate contexts, but it is also possible to use closing braces (}) to navigate.

The behavior of an explicit closing brace depends on the contents of the current command line. If the command line contains an explicit opening brace, the closing brace exits to the parent context of the opening brace.

Example

In the following example with an opening brace on the command line, the closing brace exits VPRN 1, and then enters the context of VPRN 2.

```
(ex)[/]
A:admin@node-2# configure service vprn 1 { interface "intf1" description "vprn-if" } vprn
 2

*(ex)[/configure service vprn "2"]
A:admin@node-2#
```

Example

In the following example without an opening brace on the command line, the first closing brace exits interface "int1", and the second closing brace exits VPRN 1 and enters the VPRN 2 context.

```
*[ex:/configure service]
A:admin@node-2# vprn 1 interface "int1" description "vprn-if" } } vprn 2

*[ex:/configure service vprn "2"]
A:admin@node-2#
```

2.15 Executing commands from a file

The **exec** command executes commands from a file as if the user typed or pasted the input into the MD-CLI.

The **exec** command:

- errors if it detects an interactive input
- terminates in the CLI engine in which it completes execution as follows:
  - If there are no commands that switch CLI engines, the CLI engine is always the one in which **exec** started.
  - If there are commands that switch CLI engines, **exec** ends in the last CLI engine that was entered.
  - //**exec** returns to the engine in which it was started.
- may error when used with file redirect (**>**) or output modifiers (**|**)
- terminates execution and displays an error message if an error occurs, leaving the session in the same context as when the error occurred

The system executes the file as follows:

- disables pagination while the command is running
- disables command completion while the command is running
- suppresses the commands in the file from the command history

Note: The **exec** command should not be used to execute configuration files. Use the **load** command instead.

2.15.1 Using commands that switch engines in an executable file

When using commands that switch between CLI engines within an executable file, the following commands are recommended:

- Use **/!classic-cli** to switch explicitly to the classic CLI engine and **/!md-cli** to switch explicitly to the MD-CLI engine, instead of **//** to toggle between engines.
- Use **exit all** to get to a known starting point: the operational root of the classic CLI or the MD-CLI engine.
- Include **edit-config** if the script needs to change the candidate configuration in the MD-CLI engine. Use **quit-config** after changes are committed in the script.

Note:

- An executable with **edit-config** may fail if other users have locked the configuration.
- Issuing the **quit-config** command with changes in the candidate configuration while the session is in exclusive configuration mode fails the executable because of the "discard changes" prompt.

2.16 Using paths in commands

As described in Navigating hierarchy levels, the **pwc** command displays the present working context. Navigating around the MD-CLI hierarchy changes the present working context.

The MD-CLI path format, also known as cli-path, is displayed in the MD-CLI user prompt.

Example: MD-CLI path format

```
[ex:/configure card 1 mda 1]
A:admin@node-2# pwc
Present Working Context:
  configure
  card 1
  mda 1
```

For the following commands, an absolute or relative MD-CLI path can be specified to provide path qualified attributes:

- **admin show configuration** (see Displaying configuration with admin commands)
- **annotate** (see Commenting configuration elements)
- **compare** (see Using the compare outputs to copy and paste)
- **copy** (see Using the copy command)
- **delete** (see Deleting configuration elements)
- **discard** (see Discarding configuration changes)
- **info** (see Using the info command)
- **insert** (see User-ordered lists)
- **rename** (see Using the rename command)
- **replace** (see Replacing configuration elements)
- **tree** (see Using the tree command)

The CLI path is accepted as an unnamed last parameter of the command. The information is displayed for the specified path.

Example: Information displayed for a specified CLI path

```
[ex:/configure]
A:admin@node-2# info candidate detail units /configure system
 ## apply-groups
 ## apply-groups-exclude
 ## contact
    name "node-2"
 ## location
    icmp-vse false
    selective-fib false
 ## coordinates
 ## clli-code
    ospf-dynamic-hostnames false

---snip---
```

A configuration lock from either implicit or explicit configuration mode is required to display paths for a configuration region.

The **state** branch does not have a lock and can always be specified.

Example: state branch

```
[ex:/configure]
A:admin@node-2# info detail /state system
    oper-name "node-2"
```

```
    base-mac-address aa:bb:cc:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
 ## kernel-version
 ## fabric-speed
    temperature-status ok
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
```

Example: info command with a state path

The following example shows the use of the **info** command with a **state** path from operational mode.

```
[/]
A:admin@node-2# info /state system
    oper-name "node-2"
    base-mac-address aa:bb:cc:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
    temperature-status ok
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
```

Example: Establishing a configuration lock

In the next example, to use the **info** command in a **state** context to a configuration region, a configuration lock must be established.

```
[/state system]
A:admin@node-2# info /configure system
                     ^^^^^^^^^
MINOR: MGMT_CORE #2203: Invalid element - currently not allowed

[/state system]
A:admin@node-2# edit-config read-only
INFO: CLI #2066: Entering read-only configuration mode

(ro)[/state system]
A:admin@node-2# info /configure system
    name "node-2"
    management-interface {
        configuration-mode mixed
        snmp {
            admin-state disable
        }

---snip---
```

Example: Supported commands with a CLI path parameter

```
(ex)[/state router "Base" bgp]
A:admin@node-2# info /state router bgp statistics routes-per-family ipv4
    remote-routes 45
    remote-active-routes 6
    backup-routes 0

*(ex)[/state router "Base" bgp]
A:admin@node-2# compare /configure system
-   name "test1"
```

```
+   name "test"

(ex)[/configure router "Base" bgp]
A:admin@node-2# discard /configure system

(ex)[/configure router "Base"]
A:admin@node-2# tree bgp group "mesh" add-paths ipv4
+-- receive
+-- send

[/]
A:admin@node-2# admin show configuration /configure router isis interface "system"
    passive true

(ex)[/configure policy-options]
A:admin@node-2# copy policy-statement "mytest" to /configure policy-options policy-
statement "my_new_policy"

(ex)[/configure router "Base" bgp]
A:admin@node-2# rename /configure policy-options policy-statement "mytest" to "another_
new_policy"
```

2.16.1 Absolute path

An absolute CLI path is specified using the slash (/) as the MD-CLI tree structure from the root.

Example: CLI path references a section of the MD-CLI tree under the indicated context

```
/configure card 1 mda 1
```

Example: CLI path references a specific leaf in the tree

```
/state system oper-name
```

2.16.2 Relative path

A relative CLI path specifies the MD-CLI tree structure from the present working context.

Example

In the following example, the two **info** commands display the same information. The first command usage is an absolute path reference and the next usage is a relative path from the present working context (**pwc**).

```
[ex:/configure router "Base" bgp]
A:admin@node-2# pwc
Present Working Context:
  configure
  router "Base"
  bgp

[ex:/configure router "Base" bgp]
A:admin@node-2# info /configure router "Base" bgp group "grp1"

[ex:/configure router "Base" bgp]
A:admin@node-2# info group "grp1"
```

2.17 Using expressions in commands

Several commands accept a path. For these commands, use expressions to specify multiple list keys in one command instead of entering multiple commands. For example, expressions can be used with the **info** command to display configuration or state information, and in configuration statements to enter parameter values. The following tables lists the expressions that can be used with each command.

Table 8: Using expressions in commands

Command	Wildcard	Range	Regular expression
admin show configuration	✔	✔	✔
annotate			
compare			
copy			
delete	✔	✔	✔
discard	✔	✔	✔
info	✔	✔	✔
insert	✔	✔	✔
rename			
replace			✔

The wildcard asterisk (*) character can be entered to specify all list keys, for example **info port *** or **configure port * admin-state enable**.

To specify a range of numbers that is expanded sequentially into list keys, enter a numerical range enclosed in brackets ([]). Ranges entered with numbers separated by two periods (..) are incremented by 1 when expanded, for example **port 1/1/[1..5]**. A comma (,) can be used to enter multiple numbers or ranges.

The following table lists some example ranges and how they are expanded. Numbers are processed in the order they are entered; for example, a range of [3,6] expands to 3 and 6, and a range of [6,3] expands to 6 and 3. Ranges can be at any place in a string, for example **vprn[1..3]**.

Table 9: Example range and expansion

Range	Expansion
[1..5]	1 2 3 4 5
[3,6]	3 6
[6,3]	6 3

Range	Expansion
[1..5,9]	1 2 3 4 5 9
[1..3,6,8..10]	1 2 3 6 8 9 10

A range can have a backreference to a previous range in the same command line. The reference has the form **[$n]**

where **n** is a single digit from 0 to 9 and 0 is the first match.

For example:

- **configure service vprn [1..5] router-id 10.20.[$0].1** expands to:
  - **configure service vprn 1 router-id 10.201.1.1**
  - **configure service vprn 2 router-id 10.201.2.1**
  - **configure service vprn 3 router-id 10.201.3.1**
  - **configure service vprn 4 router-id 10.201.4.1**
  - **configure service vprn 5 router-id 10.201.5.1**
- **configure router interface int[1..4] ipv6 address 2001:db8:[$0]:[1..10]::1 prefix-length 64** expands to:
  - four interfaces named **int1**, **int2**, **int3**, **int4**, each with 10 IPv6 addresses

To specify a set of strings that is expanded sequentially into list keys, enter an alphabetical sequence enclosed in brackets ([]). Strings must be separated by a comma (,), and enclosed in quotes (“) if they contain spaces. For example, **configure router interface [one,“second int”,three]** specifies interfaces named “one”, “second int”, and “three”.

To specify a matching condition for list keys, enter a regular expression enclosed in apostrophes (‘). See Using regular expressions for more information. For example, if there are three interfaces named “OLD1”, “OLD2”, and “OLD3”, **delete router interface 'OLD.*'** expands to:

- **delete router interface OLD1**
- **delete router interface OLD2**
- **delete router interface OLD3**

Note: Expressions are not supported in configuration files that are loaded at boot or using the **load** command. Wildcard deletion is supported only for leaf-lists (**delete vrf-import ***) and not for lists (**delete mld-import ***).

2.18 Using the command history

The MD-CLI command history records commands executed exactly as they were entered after **Enter** is pressed. The history can be shown with the **history** command and navigated to display each command in the history. It can also be accessed, and commands can be edited, using recall, substitution, display, and backward search.

Each MD-CLI session records its history, which cannot be shown or accessed by users in other sessions. To change the size of the history from the default of 50, set the value using the following command in the global environment for all users:

```
configure system management-interface cli md-cli environment history size
```

Alternatively, users can set the value of the **environment history size** command in the per-session environment. A size of 0 disables the history.

Table 10: History help and navigation commands

Action	Command
Show the command history	**history**
Show all commands in the history	**history all**
Show the commands in the history	**history commands-only**
Show the help	**!?**
Move forward	**Ctrl-N**, Down arrow
Move backward	**Ctrl-P**, Up arrow

The history is context aware so that it displays and accesses operational mode commands and commands entered in a configuration mode in each region separately. For example, commands entered in a configuration mode are not available in the operational mode history. The following table describes the availability of commands in the history.

Table 11: Availability of history commands

Mode	Command History
Operational	Operational commands **state** commands
configure bof or **edit-config bof**	Operational commands **state** commands Configuration mode commands **bof** region commands
configure or **edit-config**	Operational commands **state** commands Configuration mode commands **configure** region commands
configure debug or	Operational commands **state** commands Configuration mode commands

Mode	Command History
edit-config debug	**debug** region commands

Example: Show the context-aware history

The **history** command output in this example shows configuration mode and configuration commands in the first output but does not in the second output in operational mode.

```
[ex:/configure router "Base" interface "NEW INTERFACE"]
A:admin@node-2# history

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
     3 10:30 info state system bootup last-boot-config-version
     4 10:31 configure exclusive
     5 10:31 router interface "NEW INTERFACE"
     6 10:31 commit
     7 10:31 history

[ex:/configure router "Base" interface "NEW INTERFACE"]
A:admin@node-2# exit all

[/]
A:admin@node-2# history

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
     3 10:30 info state system bootup last-boot-config-version
     4 10:31 configure exclusive
     7 10:31 history
     8 10:31 exit all
     9 10:32 history
```

Example: Show all commands in the history

The **history all** command shows all commands in the history, regardless of the current mode. This example shows configuration mode and configuration commands in operational mode.

```
[/]
A:admin@node-2# history all

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
     3 10:30 info state system bootup last-boot-config-version
     4 10:31 configure exclusive
     5 10:31 router interface "NEW INTERFACE"
     6 10:31 commit
     7 10:31 history
     8 10:31 exit all
     9 10:32 history all
```

Example: Show commands in the history

To show the history without the entry numbers and execution times for copying and pasting, use the **history commands-only** command.

```
[/]
A:admin@node-2# history commands-only
```

```
show version
ping 192.168.0.10 count 1
info state system bootup last-boot-config-version
configure exclusive
history
exit all
history commands-only
```

Note: The **all** and **commands-only** commands can be combined.

Users can access the history for execution or command editing using several methods. To disable history recall, substitution, display, and backward search, set the following command in the global environment for all users:

```
configure system management-interface cli md-cli environment history recall false
```

Alternatively, users can set the **environment history recall false** command in the per-session environment.

Table 12: History access commands available for use

Action	Command
Recall and execute by entry number	**!***number* [*element*...]
Recall and execute by string search	**!***string* [*element*...]
Recall and execute previous command	**!!** [*element*...]
Substitute last element	**!$**
Display entry number	**!***number***:p**
Display entry with string search	**!***string***:p**
Display previous command	**!!:p**
Display last element	**Esc+.**
Backward search	**Ctrl-R**
Exit search	**Ctrl-C**

Recall and execute the history by entering **!** (exclamation point) followed by an entry number, a string search, **!** for the previous command, or **$** to substitute the last element that was entered in the previous command. The following history is used at the beginning of each of the following examples.

```
[/]
A:admin@node-2# history

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
```

Example: Recall by entry number

```
[/]
A:admin@node-2# !2

[/]
A:admin@node-2# ping 192.168.0.10 count 1
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=121 time=1.58ms.

---- 192.168.0.10 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
round-trip min = 1.58ms, avg = 1.58ms, max = 1.58ms, stddev = 0.000ms
```

Example: Recall by search string

```
[/]
A:admin@node-2# !192

[/]
A:admin@node-2# ping 192.168.0.10 count 1
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=121 time=1.39ms.

---- 192.168.0.10 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
round-trip min = 1.39ms, avg = 1.39ms, max = 1.39ms, stddev = 0.000ms

[/]
A:admin@node-2# !ping

[/]
A:admin@node-2# ping 192.168.0.10 count 1
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=121 time=1.27ms.

---- 192.168.0.10 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
round-trip min = 1.27ms, avg = 1.27ms, max = 1.27ms, stddev = 0.000ms
```

Example: Recall previous command

```
[/]
A:admin@node-2# !!

[/]
A:admin@node-2# ping 192.168.0.10 count 1
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=121 time=1.29ms.

---- 192.168.0.10 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
round-trip min = 1.29ms, avg = 1.29ms, max = 1.29ms, stddev = 0.000ms
```

Example: Substitute previous element

```
[/]
A:admin@node-2# ping 192.168.0.10
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=114 time=120ms.
64 bytes from 192.168.0.10: icmp_seq=2 ttl=114 time=120ms.
```

```
64 bytes from 192.168.0.10: icmp_seq=3 ttl=114 time=120ms.
64 bytes from 192.168.0.10: icmp_seq=4 ttl=114 time=120ms.
64 bytes from 192.168.0.10: icmp_seq=5 ttl=114 time=120ms.

---- 192.168.0.10 PING Statistics ----
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min = 120ms, avg = 120ms, max = 120ms, stddev = 0.091ms

[/]
A:admin@node-2# traceroute !$

[/]
A:admin@node-2# traceroute 192.168.0.10
traceroute to 135.92.55.3, 30 hops max, 40 byte packets
  1  10.0.0.1 (10.0.0.1)    0.948 ms  0.754 ms  0.825 ms
  2  192.168.0.10 (192.168.0.10)    1.14 ms  1.08 ms  9.92 ms
```

Example: Recall and enter additional elements

Additional elements can be entered after recalled commands to add to the command line. ? help and Tab command completion are supported after recalled commands. In this example, the previous **ping** command is recalled and **size 1500** is added.

```
[/]
A:admin@cses-V93# !ping size 1500

[/]
A:admin@node-2# ping 192.168.0.10 count 1 size 1500
PING 192.168.0.10 1500 data bytes
1508 bytes from 192.168.0.10: icmp_seq=1 ttl=63 time=3.30ms.

---- 135.92.55.3 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
round-trip min = 3.30ms, avg = 3.30ms, max = 3.30ms, stddev = 0.000ms
```

Example

The history can be displayed to edit the command before execution by entering **!** (exclamation mark) followed by an entry number, a string search, or **!** for the previous command followed by **:p**. **Esc+.** substitutes the last element that was entered in the previous command. The following history is used at the beginning of each of these examples, and the □ character indicates the cursor position after the command is displayed.

```
[/]
A:admin@node-2# history

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
```

Example: Display entry number

```
[/]
A:admin@node-2# !2:p

[/]
A:admin@node-2# ping 192.168.0.10 count 1□
```

Example: Display entry with string search

```
[/]
A:admin@node-2# !192:p

[/]
A:admin@node-2# ping 192.168.0.10 count 1□

[/]
A:admin@node-2# !ping:p

[/]
A:admin@node-2# ping 192.168.0.10 count 1□
```

Example: Display previous command

```
[/]
A:admin@node-2# !!:p

[/]
A:admin@node-2# ping 192.168.0.10 count 1□
```

Example: Display last element

```
[/]
A:admin@node-2# traceroute 192.168.0.10□  Esc+. was entered and "192.168.0.10" is
 displayed
```

The history can be searched backward interactively with a string search when **Ctrl-R** is entered. Additional **Ctrl-R** keystrokes search backward again, and `(wrapped)` is displayed when the search wraps. Pressing **Enter** exits search and executes the command, and **Ctrl-C** exits search without execution. The following history is used at the beginning of the following example.

```
[/]
A:admin@node-2# history

     1 10:30 show version
     2 10:30 ping 192.168.0.10 count 1
```

Example: Backward search

```
[/]
History search '':  Entering Ctrl-R displays the search prompt

[/]
History search 'pi': ping 192.168.0.10 count 1  Entering "pi" displays the first match

[/]
History search 'pi' (wrapped): ping 192.168.0.10 count 1  Entering Ctrl-R again indicates
 the search wrapped

[/]
A:admin@node-2# ping 192.168.0.10 count 1  Enter exits search and executes the command
PING 192.168.0.10 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=1 ttl=121 time=1.29ms.

---- 192.168.0.10 PING Statistics ----
1 packet transmitted, 1 packet received, 0.00% packet loss
```

```
round-trip min = 1.29ms, avg = 1.29ms, max = 1.29ms, stddev = 0.000ms
```

3 Displaying configuration and state information

3.1 Using the info command

The **info** command shows configuration or state information from the present context. The command can only be executed in a configuration mode for a configuration region or for the state branch. By default, all configured parameters in the candidate configuration datastore are displayed for a configuration region. For a state region, all elements that have a value are displayed.

Table 13: Displaying configuration from configuration groups and third-party models with info options

Usage	Info option	Datastore	Nokia configuration	Configuration groups	Third-party model configuration
Display third-party configuration converted to Nokia configuration	**converted**	Running	Nokia and converted third-party configuration with origin comments	**groups** and unexpanded **apply-groups** statements	—
Display the configuration with configuration groups inheritance	**inheritance**	Any	✓	**groups** and expanded **apply-groups** statements with origin comments	Native format
Display third-party configuration converted to Nokia configuration with configuration groups inheritance	**converted inheritance**	Running	Nokia and converted third-party configuration with origin comments	**groups** and expanded **apply-groups** statements with origin comments	—
Display the expanded configuration applied by the router	**intended**	Intended	✓	Expanded configuration without **groups** or **apply-groups** statements or origin comments	Native format

Note: The **converted** and **model** options are only available when **configure system management-interface yang-modules openconfig-modules** is set to **true**.

Note: The output of **info flat** and **info full-context** displays braces ({}) relative to the context from which the command is executed so that the session remains in the context if the output is pasted. Braces are displayed around empty containers that enable functionality and around lists

that cannot be entered for future compatibility, in case elements are added to the list. Execute the command from the configuration root to stay at the root context without navigating.

Note: As indicated in the choice statement, the **flat**, **json**, and **xml** options are mutually exclusive options. Other unsupported combinations include:

- [**from**] **candidate** and **converted**
- [**from**] **intended** and **converted**
- [**from**] **intended** and **detail**
- [**from**] **intended** and **inheritance**
- **full-context** and **flat**
- **model** and **detail**
- **model** and **inheritance**
- **units** and **json**
- **units** and **xml**

The order of the configuration output is as follows:

- Keys are displayed on the same line as the command element.
- **apply-groups** is displayed, if applicable.
- **apply-groups-exclude** is displayed, if applicable.
- **admin-state** is displayed, if applicable.
- **description** is displayed, if applicable.
- Other top-level elements are displayed in alphabetical order.

Example: Displaying configured information

The following displays configured information for **configure router bgp**.

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info
    connect-retry 90
    local-preference 250
    add-paths {
        ipv4 {
            receive true
        }
    }
```

The following output displays the same information in JSON IETF format.

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info json
{
    "nokia-conf:connect-retry": 90,
    "nokia-conf:local-preference": 250,
    "nokia-conf:add-paths": {
        "ipv4": {
            "receive": true
        }
    }
}
```

The following output displays the same information in XML format:

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info xml
<connect-retry xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">90</connect-retry>
<local-preference xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">250</local-preference>
<add-paths xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
    <ipv4>
        <receive>true</receive>
    </ipv4>
</add-paths>
```

The configuration output can display all elements that are configured, even if an element is set to the system default state or value. The **detail** option displays all data for the context, including default configurations. The double hash (##) indicates an unconfigured element or a dynamic default.

Example: Displaying all configured elements

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info detail
 ## apply-groups
    admin-state enable
 ## description
    connect-retry 90
    keepalive 30
    damping false
    local-preference 250
    loop-detect ignore-loop
    loop-detect-threshold 0
    selective-label-ipv4-install false
    min-route-advertisement 30
    aggregator-id-zero false
    preference 170
    block-prefix-sid false
 ## multihop
 ## med-out
 ## authentication-key
    client-reflect true
    vpn-apply-export false
    vpn-apply-import false
    asn-4-byte true
```

When using the **info** command with both the **detail** and **xml** options, the double hash (##) elements (indicating unconfigured elements or dynamic defaults) are enclosed within XML comments.

Example: Using the detail and xml options

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info detail xml
<!-- ## apply-groups -->
<!-- ## apply-groups-exclude -->
<admin-state xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">enable</admin-state>
<!-- ## description -->
<connect-retry xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">90</connect-retry>
<keepalive xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">30</keepalive>
```

```
<damping xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">false</damping>
<local-preference xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">250</local-preference>

...
```

When using the **info** command with both **detail** and **json** options, the output does not include unconfigured elements. Unconfigured elements in the MD-CLI are denoted with ## and there is no standard method of displaying comments within the JSON format.

Example: Using the detail and json options

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info detail json
{
    "nokia-conf:admin-state": "enable",
    "nokia-conf:connect-retry": 90,
    "nokia-conf:keepalive": 30,
    "nokia-conf:damping": false,
    "nokia-conf:local-preference": 250,
    "nokia-conf:loop-detect": "ignore-loop",
    "nokia-conf:loop-detect-threshold": 0,
    "nokia-conf:selective-label-ipv4-install": false,
    "nokia-conf:min-route-advertisement": 30,
    "nokia-conf:aggregator-id-zero": false,
    "nokia-conf:preference": 170,
    "nokia-conf:block-prefix-sid": false,
    "nokia-conf:client-reflect": true,
    "nokia-conf:vpn-apply-export": false,
    "nokia-conf:vpn-apply-import": false,
    "nokia-conf:asn-4-byte": true,
    "nokia-conf:path-mtu-discovery": false,
    "nokia-conf:enforce-first-as": false,
    "nokia-conf:initial-send-delay-zero": false,
    "nokia-conf:inter-as-vpn": false,
    "nokia-conf:purge-timer": 10,
    "nokia-conf:route-table-install": true,

...
```

The **flat** option displays the context of every element in the present working context on a single line. Braces ensure that the context stays in the present working context for copy and paste purposes.

Example: Using the flat option

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info flat
connect-retry 90
    local-preference 250
    add-paths ipv4 receive true
```

The **full-context** option displays the full context of every element from the present working context on a single line.

Example: Using the full-context option

```
*[ex:/configure router "Base" bgp]
A:admin@cses-V93# info full-context
    /configure router "Base" bgp connect-retry 90
    /configure router "Base" bgp local-preference 250
```

```
    /configure router "Base" bgp add-paths ipv4 receive true
```

When the **full-context** option is used in conjunction with the **json** or **xml** option, the output is modified to display the configuration back to the YANG model-aware root of the configuration tree.

Example: Using the full-context option with the json or xml option

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info full-context json
{
    "nokia-conf:configure": {
        "router": [
            {
                "router-name": "Base",
                "bgp": {
                    "connect-retry": 90,
                    "local-preference": 250,
                    "add-paths": {
                        "ipv4": {
                            "receive": true
                        }
                    }
                }
            }
        ]
    }
}
```

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info full-context xml
<configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
    <router>
        <router-name>Base</router-name>
        <bgp>
            <connect-retry>90</connect-retry>
            <local-preference>250</local-preference>
            <add-paths>
                <ipv4>
                    <receive>true</receive>
                </ipv4>
            </add-paths>
        </bgp>
    </router>
</configure>
```

The **depth** option displays configuration from the present working context limited to the specified depth. Use this option to suppress output that is lower in the hierarchy. For example, if the present working context is **configure**, the depths of the commands are:

- **configure router** = 1
- **configure router description** = 2
- **configure router interface description** = 3

Example: Using the depth option

```
[ex:/configure router "Base"]
A:admin@node-2# info depth 1
    autonomous-system 65816
    router-id 172.16.255.0
```

```
    }
    interface "system" {
    }
    bgp {
    }
    isis 0 {
    }
```

3.1.1 Displaying lists

The **info** command requires the user to enter a list key name. Users can enter the wildcard asterisk (*) character, a range, or a regular expression instead of a list key name to display a subset of keys with one command. For more information, see Using expressions in commands.

Example: Displaying lists with a list key name

```
[ex:/configure]
A:admin@node-2# info sfm 1
    sfm-type sfm-s

[ex:/configure]
A:admin@node-2# info sfm 2
    mda-type sfm-s
```

Example: Displaying lists with the wildcard * character

```
[ex:/configure]
A:admin@node-2# info sfm *
    sfm 1 {
        sfm-type sfm-s
    }
    sfm 2 {
        sfm-type sfm-s
    }
    sfm 3 {
        sfm-type sfm-s
    }
    sfm 4 {
        sfm-type sfm-s
    }
```

Example: Displaying lists with a range

```
[ex:/configure]
A:admin@node-2# info sfm [3..4]
    sfm 3 {
        sfm-type sfm-s
    }
    sfm 4 {
        sfm-type sfm-s
    }

[ex:/configure]
A:admin@node-2# info sfm [1,3]
    sfm 1 {
        sfm-type sfm-s
    }
    sfm 3 {
        sfm-type sfm-s
```

```
    }

[ex:/configure]
A:admin@node-2# info port 1/1/[1..3]
    port 1/1/1 {
        admin-state enable
    }
    port 1/1/2 {
        admin-state enable
    }
    port 1/1/3 {
        admin-state enable
   }
```

Example: Displaying lists with a regular expression

```
[ex:/configure router "Base"]
A:admin@node-23# info interface 'sys'
 interface "system" {
     ipv4 {
         primary {
             address 172.16.255.0
             prefix-length 32
         }
     }
  }
```

The **info** command always displays all keys of the list on the same line. The first key of a list is unnamed in the MD-CLI, however, there are exceptions where the key is named and must be entered. (See the online help for the correct command syntax , or the *7705 SAR Gen 2 MD-CLI Command Reference Guide*). All other keys are named.

Example

For example, the **collector** list has two keys, **ip-address** and **port**. The name of the first key, **ip-address**, does not appear in the **info** display. The name of the second key and any subsequent keys are always displayed.

```
*[ex:/configure cflowd]
A:admin@node-2# info
    collector 10.10.20.30 port 7 {
    }
    collector 10.10.30.40 port 8 {
    }
```

3.2 Using operational commands

The **clear**, **monitor**, **show**, and **tools** commands in the MD-CLI display the same information and provide the same functionality as they do in the classic CLI. No additional outputs or enhancements are included in the MD-CLI. See the *7705 SAR Gen 2 Clear, Monitor, Show, Tools CLI Command Reference Guide* for more information about the SR OS CLI commands that are used to manage the SR OS in either CLI engine.

The information in the output of the **show** commands can also be found in the **state** and **configure** branches of the MD-CLI (as well as via model-driven interfaces such as NETCONF and gRPC). The **show** commands, however, are different from the YANG state output.

The YANG state is fully modeled and structured data that can be easily manipulated by tools and applications (including pySROS and NETCONF clients). The report-style output of the **show** commands contains useful combinations of configuration and state, but the information is not modeled or structured. The pySROS and MD-CLI command alias features can be used together to create custom show commands using the modeled data. See Command aliases for more information.

Note: Follow the classic CLI context when using the **show** command. For example, route policy information is displayed using the **show router policy** command in both the MD-CLI and classic CLI engines, even though this information is configured in the **configure policy-options** context in the MD-CLI and in the **configure router policy-options** context in the classic CLI.

The following classic CLI **show** commands are not available in the MD-CLI and have equivalent commands:

- **show alias**

  Use the MD-CLI **info detail environment command-alias** command.
- **show bof**

  Use the MD-CLI **admin show configuration bof** command or the **info** command in the bof configuration region.
- **show debug**

  Use the MD-CLI **admin show configuration debug** command or the **info** command in the debug configuration region.
- **show system candidate**

  Use the MD-CLI **info** command in a configuration mode.
- **show system rollback**

  Use the MD-CLI **show system management-interface commit-history** command.

3.3 Displaying state information

The same navigation commands used in a configuration region are available in the MD-CLI state tree. Online ? help and command completion functionality are also available. The state tree information can be displayed using the **info** command.

Note: The **converted**, **inheritance**, and **model** options are not supported for **state** elements.

Example: Output displays information from the state tree

The following output displays information from the state tree using the **info** command, with the time format defined in RFC 1123.

```
[/environment]
A:admin@node-2# time-format rfc-1123

[/environment]
A:admin@node-2# state system

[/state system]
A:admin@node-2# info
```

```
    oper-name "node-2"
    base-mac-address aa:bb:cc:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
    temperature-status ok
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
    up-time 98563010
    current-time "Thu, 10 Apr 2025 16:44:33 UTC"
    reboot-required false
    reboot-required-reason []
    os-security-password disabled
    anti-theft-locked false
    boot-good-exec-status success
    boot-bad-exec-status not-run
    alarms {
    }
    grpc {
        oper-state down
        supported-services {
            gnmi-version "0.7.0"
            gnoi-cert-mgmt-version "0.1.0"
            gnoi-system-version "0.1.0"

---snip---
```

Example: Using the units option of the info command

The **units** option of the **info** command displays the output with unit types for applicable elements.

```
[/state system]
A:admin@node-2# info units
    oper-name "node-2"
    base-mac-address aa:bb:ff:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
    temperature-status ok
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
    up-time 98667790 milliseconds
    current-time "2025-04-10 16:46:18 UTC"
    reboot-required false
    reboot-required-reason []
    os-security-password disabled
    anti-theft-locked false
    boot-good-exec-status success
    boot-bad-exec-status not-run
    alarms {
    }
    grpc {
        oper-state down
        supported-services {
            gnmi-version "0.7.0"
            gnoi-cert-mgmt-version "0.1.0"
            gnoi-system-version "1.0.0"
            md-cli-version "0.1.0"
            rib-api-version "1.1.0"
        }
    }
    management-interface {
        configuration-oper-mode model-driven
```

```
        last-mode-switch "2025-04-9 23:27:47 UTC"
        last-mode-switch-duration 12 milliseconds

---snip---
```

Example: Using the json option of the info command

The **json** option of the **info** command displays the output in indented JSON IETF format.

```
[/state system]
A:admin@node-2# info json
{
    "nokia-state:oper-name": "node-2",
    "nokia-state:base-mac-address": "aa:bb:ff:00:00:00",
    "nokia-state:platform": "7705 SAR-1",
    "nokia-state:chassis-topology": "standalone",
    "nokia-state:temperature-status": "ok",
    "nokia-state:fp-generation-vfp": true,
    "nokia-state:system-profile": "profile-a",
    "nokia-state:active-cpm-slot": "A",
    "nokia-state:up-time": "98714800",
    "nokia-state:current-time": "2025-04-10 16:47:05 UTC",
    "nokia-state:reboot-required": false,
    "nokia-state:reboot-required-reason": "",
    "nokia-state:os-security-password": "disabled",
    "nokia-state:anti-theft-locked": false,
    "nokia-state:boot-good-exec-status": "success",
    "nokia-state:boot-bad-exec-status": "not-run",
    "nokia-state:alarms": {
    },
    "nokia-state:grpc": {
        "oper-state": "down",
        "supported-services": {
            "gnmi-version": "0.7.0",
            "gnoi-cert-mgmt-version": "0.1.0",
            "gnoi-system-version": "1.0.0",
            "md-cli-version": "0.1.0",
            "rib-api-version": "1.1.0"
        }
    },
    "nokia-state:management-interface": {
        "configuration-oper-mode": "model-driven",
        "last-mode-switch": "2025-04-9 23:27:47 UTC",
        "last-mode-switch-duration": 12,

---snip---
```

Example: Output using the time format defined in RFC 3339 (default output format)

```
[/]
A:admin@node-2# environment time-format rfc-3339

[/]
A:admin@node-2# state system

[/state system]
A:admin@node-2# info
    oper-name "node-2"
    base-mac-address aa:bb:cc:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
    temperature-status ok
```

```
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
    up-time 98377080
    current-time 2025-04-10T16:41:27.8+00:00
    reboot-required false
    reboot-required-reason []
    os-security-password disabled
    anti-theft-locked false
    boot-good-exec-status success
    boot-bad-exec-status not-run
    alarms {
    }
    grpc {
        oper-state down
        supported-services {
            gnmi-version "0.7.0"
            gnoi-cert-mgmt-version "0.1.0"
            gnoi-system-version "0.1.0"

---snip---
```

Example: Output using the time format defined in ISO 8601

```
[/]
A:admin@node-2# environment time-format iso-8601

[/]
A:admin@node-2# state system

[/state system]
A:admin@node-2# info
    oper-name "node-2"
    base-mac-address aa:bb:cc:00:00:00
    platform "7705 SAR-1"
    chassis-topology standalone
    temperature-status ok
    fp-generation-vfp true
    system-profile profile-a
    active-cpm-slot "A"
    up-time 98417520
    current-time "2025-04-10 16:42:08 UTC"
    reboot-required false
    reboot-required-reason []
    os-security-password disabled
    anti-theft-locked false
    boot-good-exec-status success
    boot-bad-exec-status not-run
    alarms {
    }
    grpc {
        oper-state up
        supported-services {
            gnmi-version "0.8.0"
            gnoi-cert-mgmt-version "0.1.0"
            gnoi-file-version "0.1.0"
            gnoi-system-version "1.0.0"
            md-cli-version "25.3.0"
        }
    }

---snip---
```

3.4 Displaying configuration with admin commands

The **admin show** commands display the same configuration as the **info** command but are not subject to command authorization and do not require configuration mode access.

See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for information about the **admin** commands in the MD-CLI.

Example

The following examples show the **admin show configuration** output of the running configuration in the default configuration region (**configure**). Other configuration regions can also be specified, including **bof**, **debug**, and **li**.

```
[/]
A:admin@node-2# admin show configuration
# TiMOS-B-25.3.B1-23 both/x86_64 Nokia 7705 SAR Copyright (c) 2000-2025 Nokia.
# All rights reserved. All use subject to applicable license agreements.
# Built on Tue Apr 8 01:14:58 UTC 2025 by builder in /builds/253B/B1-23/panos/main/srux
# Configuration format version 25.3 revision 0

# Generated Thu Apr 10 16:49:00 2025 UTC

configure {
    card 1 {
        mda 2 {
            mda-type isa-tunnel-v
        }
        mda 3 {
            mda-type isa-bb-v
        }
    }
    log {
        filter "1001" {
            named-entry "10" {
                description "Collect only events of major severity or higher"
                action forward
                match {
                    severity {
                        gte major
                    }
                }
            }
        }
        log-id "90" {
            source {
                main true
                change true
            }
            destination {
                snmp {
                }
            }
        }
        log-id "99" {
            description "Default System Log"
            source {
                main true
            }
            destination {
                memory {
```

```
                    max-entries 500
                }
            }
        }
        log-id "100" {
            description "Default Serious Errors Log"
            filter "1001"
            source {
                main true
            }
            destination {
                memory {
                    max-entries 500
                }
            }
        }
        snmp-trap-group "90" {
            trap-target "192.168.135.228:162" {
                address 192.168.135.228
                version snmpv2c
                notify-community "private"
            }
        }
    }


...
```

```
[/]
A:admin@node-2# admin show configuration full-context
# TiMOS-B-25.3.B1-23 both/x86_64 Nokia 7705 SAR Copyright (c) 2000-2025 Nokia.
# All rights reserved. All use subject to applicable license agreements.
# Built on Tue Apr 8 01:14:58 UTC 2025 by builder in /builds/253B/B1-23/panos/main/srux
# Configuration format version 25.3 revision 0

# Generated Thu Apr 10 16:49:53 2025 UTC

    /configure card 1 mda 2 mda-type isa-tunnel-v
    /configure card 1 mda 3 mda-type isa-bb-v
    /configure log filter "1001" named-entry "10" description "Collect only events of
 major severity or higher"
    /configure log filter "1001" named-entry "10" action forward
    /configure log filter "1001" named-entry "10" match severity gte major
    /configure log log-id "90" source main true
    /configure log log-id "90" source change true
    /configure { log log-id "90" destination snmp }
    /configure log log-id "99" description "Default System Log"
    /configure log log-id "99" source main true
    /configure log log-id "99" destination memory max-entries 500
    /configure log log-id "100" description "Default Serious Errors Log"
    /configure log log-id "100" filter "1001"
    /configure log log-id "100" source main true
    /configure log log-id "100" destination memory max-entries 500


...
```

```
[/]
A:admin@node-2# admin show configuration json
{
    "nokia-conf:configure": {
        "card": [
```

```
            {
                "slot-number": 1,
                "mda": [
                    {
                        "mda-slot": 2,
                        "mda-type": "isa-tunnel-v"
                    },
                    {
                        "mda-slot": 3,
                        "mda-type": "isa-bb-v"
                    }
                ]
            }
        ],
        "log": {
            "filter": [
                {
                    "filter-name": "1001",
                    "named-entry": [
                        {
                            "entry-name": "10",
                            "description": "Collect only events of major severity or
 higher",
                            "action": "forward",
                            "match": {
                                "severity": {
                                    "gte": "major"
                                }
                            }
                        }
                    ]
                }
            ],
            "log-id": [
                {
                    "name": "90",
                    "source": {
                        "main": true,
                        "change": true
                    },
                    "destination": {
                        "snmp": {
                        }
                    }
                },
                {
                    "name": "99",
                    "description": "Default System Log",
                    "source": {
                        "main": true
                    },
                    "destination": {
                        "memory": {
                            "max-entries": 500
                        }
                    }
                }
            ]
        },

...
```

```
[/]
```

```
A:admin@node-2# admin show configuration xml
<configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nc=
"urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
    <card>
        <slot-number>1</slot-number>
        <mda>
            <mda-slot>2</mda-slot>
            <mda-type>isa-tunnel-v</mda-type>
        </mda>
        <mda>
            <mda-slot>3</mda-slot>
            <mda-type>isa-bb-v</mda-type>
        </mda>
    </card>
    <log>
        <filter>
            <filter-name>1001</filter-name>
            <named-entry>
                <entry-name>10</entry-name>
                <description>Collect only events of major severity or higher</description>

...
```

4 Editing configuration

4.1 Configuration workflow

4.1.1 MD-CLI session modes

There are two modes in the MD-CLI:

- **operational**

  A user can run commands to monitor or troubleshoot the router but cannot enter configuration mode. The router configuration cannot be changed and must be displayed with administrative commands.

- **configuration**

  A user can run commands to monitor or troubleshoot the router and can enter configuration mode. In private, exclusive, or global configuration mode, the router configuration can be changed. In read-only configuration mode, the user can only view the router configuration.

The first line of the user prompt indicates the active configuration mode. For example:

- [**pr**:configure]

  This indicates a user in private configuration mode (implicit configuration workflow).

- (**ex**) [configure]

  This indicates a user in exclusive configuration mode (explicit configuration workflow).

At login, an MD-CLI session always starts in operational mode. To configure the router, the user must enter a configuration mode using the explicit or implicit configuration workflow.

The configuration workflow (implicit or explicit) determines if the user is restricted to the configure branch or if the user can navigate freely while in configuration mode. Configuration workflows are detailed in Implicit and explicit configuration workflows.

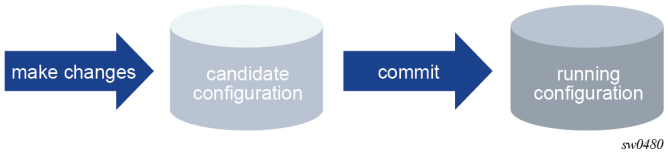
The configuration mode (private, exclusive, global, or read-only) determines the interaction with other simultaneous configuration sessions. Candidate configuration modes are detailed in Candidate configuration modes.

4.1.2 Transactional configuration method

The MD-CLI transactional configuration method is a two-step process in which configuration changes are made in a candidate configuration. When the configuration is committed, the changes are copied to the running configuration and become active.

The following figure shows the flow of configuration changes from the candidate configuration to the running configuration.

Figure 1: Flow of configuration changes



Other non-router configuration operations, such as changing the MD-CLI session environment are active immediately.

The MD-CLI configuration method differs from the classic CLI in the following ways:

- In the classic CLI, changes to the router configuration are immediately activated in the running configuration. A strict configuration order must be maintained or the configuration fails.
- In the MD-CLI, the transactional configuration method allows multiple configuration changes to be made in any order in the candidate configuration. The system applies the correct ordering when the configuration is activated with the **commit** command.

4.1.3 Implicit and explicit configuration workflows

The MD-CLI supports two configuration workflows:

- Implicit configuration workflow
  - Navigation is restricted to the **configure** branch and its descendants.
  - Operational commands require an absolute path and error when incomplete.
  - **configure** {**private** | **exclusive** | **global** | **read-only**} enters configuration mode and navigates in the **configure** branch. There is no default configuration mode.
  - **exit all** leaves configuration mode and navigates to the operational root.
- Explicit configuration workflow
  - Navigation is unrestricted while in configuration mode.
  - Operational commands while in the **configure** branch require an absolute path and navigate when incomplete.
  - **edit-config** {**private** | **exclusive** | **global** | **read-only**} enters configuration mode without navigating. There is no default configuration mode.
  - **quit-config** leaves configuration mode without navigating. The **quit-config** command is not available in the **configure** branch.

The following table compares the implicit and explicit configuration workflows.

Table 14: Implicit and explicit configuration mode features

	Implicit configuration workflow	Explicit configuration workflow
Use	User focused on configuration tasks in the **configure** branch	Power user mode with unrestricted navigation capabilities

	Implicit configuration workflow	Explicit configuration workflow
Flexibility	Run operational commands or configuration commands from the **configure** branch	Run operational commands or configuration commands anywhere
configure	Enters configuration mode[2] and navigates to the **configure** branch	Navigates to the **configure** branch (after **edit-config** command)
edit-config	—	Enters configuration mode[2]
exit all or **Ctrl-Z** or **/**	Leaves configuration mode and navigates to the operational root	Navigates to the operational root
quit-config	—	Leaves configuration mode
Commands that result in an action or display output	Execute the command	Execute the command
Commands that navigate out of the **configure** branch	Not allowed	Navigate
info and configuration commands in the **configure** branch	Allowed	Allowed
info and configuration commands out of the **configure** branch	Not allowed	Allowed

4.1.3.1 Using the implicit configuration workflow

In the implicit configuration workflow, navigation while in configuration mode is restricted to the **configure** branch and its descendants.

The **configure** {**private** | **exclusive** | **global** | **read-only**} command places the user session in the specified configuration mode and navigates to the top of the configuration tree (**/configure**).

Example: First line indicated configuration mode

The first line of the session prompt indicates the configuration mode prepended to the context and separated with a colon.

```
[/]
A:admin@node-2# configure exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit
```

[2] Requires specifying the configuration mode (private | exclusive | global | read-only)

```
[ex:/configure]
A:admin@node-2#
```

When the MD-CLI session is in operational mode, the **configure** command only accepts a configuration mode parameter and cannot be followed by a path to navigate nor by a configuration element to edit the router configuration.

Example: MD-CLI session in operational mode

```
[/]
A:admin@node-2# configure exclusive router
                                  ^^^^^^
MINOR: CLI #2069: Operation not allowed - currently in operational mode

[/]
A:admin@node-2#
```

The following navigation commands leave configuration mode if they cause navigation outside the configuration branch:

- **back**, or **back** with a number greater than the present working context depth
- **exit**, or **exit all**
- **Ctrl-Z**
- **/**
- **}**

Example: Exiting the configuration mode

```
[ex:/configure router "Base"]
A:admin@node-2# exit all
INFO: CLI #2064: Exiting exclusive configuration mode

[/]
A:admin@node-2#
```

Example: Commands that do not navigate outside the configure branch

Commands that do not navigate outside the configure branch or that result in an action or display output are allowed.

```
[ex:/configure]
A:admin@node-2# show uptime
System Up Time         : 3 days, 00:27:49.35 (hr:min:sec)

[ex:/configure]
A:admin@node-2#
```

```
[ex:configure]
A:admin@node-2# /environment more false

[ex:/configure]
A:admin@node-2#
```

Example: Commands that navigate outside the configure branch

Commands that navigate out of a configure branch are not allowed.

```
[ex:/configure]
A:admin@node-2# show router
MINOR: CLI #2069: Operation not allowed - cannot navigate out of configuration region

[ex:/configure]
A:admin@node-2#
```

```
[ex:/configure]
A:admin@node-2# tools dump
MINOR: CLI #2069: Operation not allowed - cannot navigate out of configuration region

[ex:/configure]
A:admin@node-2#
```

4.1.3.2 Using the explicit configuration workflow

In the explicit configuration workflow, navigation while in configuration mode is unrestricted. Operational and configuration commands can be executed from any context.

The **edit-config** {**private** | **exclusive** | **global** | **read-only**} command places the user session in the specified configuration mode. The present working context is not changed.

Example: Showing configuration mode in the prompt

The first line of the session prompt indicates the configuration mode between round brackets.

```
[/show router]
A:admin@node-2# edit-config exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

(ex)[/show router]
A:admin@node-2#
```

When the MD-CLI session is in configuration mode, the **configure** command can be followed by a path to navigate or by a configuration element to edit the router configuration.

Example: Command following by a path or configuration element

```
(ex)[/]
A:admin@node-2# show router

(ex)[/show router]
A:admin@node-2# /configure system time zone standard name utc

*(ex)[/show router]
A:admin@node-2# /configure router

*(ex)[/configure router "Base"]
A:admin@node-2#
```

Commands that result in an action or display output can be executed in the configure branch. Navigation outside the configure branch is allowed and does not exit the configuration mode.

Example: Navigation outside the configure branch

```
(ex)[/configure router "Base"]
A:admin@node-2# show uptime
System Up Time         : 8 days, 23:16:45.01 (hr:min:sec)

(ex)[/configure router "Base"]
A:admin@node-2# tools

(ex)[/tools]
A:admin@node-2#
```

Configuration commands, such as **info** and **commit**, can be executed outside the **configure** branch.

Example: Executing commands outside the configure branch

```
(ex)[/tools]
A:admin@node-2# info
    configure {
        log {
       {
---snip---

(ex)[/tools]
A:admin@node-2# commit

(ex)[/tools]
A:admin@node-2#
```

The **quit-config** command exits configuration mode and places the session in operational mode. The **quit-config** command must be executed from the operational root. The present working context does not change.

Example: Using the quit-config command

```
(ex)[/tools]
A:admin@node-2# exit all

(ex)[/]
A:admin@node-2# quit-config
INFO: CLI #2064: Exiting exclusive configuration mode
```

4.1.3.3 Transitioning from an implicit to an explicit configuration workflow

An MD-CLI configuration session can transition from an implicit to an explicit configuration workflow using the **edit-config** command while in configuration mode.

Transitioning from an explicit to an implicit configuration workflow is not supported.

Example

```
[/]
A:admin@node-2# configure exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

[ex:/configure]
```

```
A:admin@node-2# show
MINOR: CLI #2069: Operation not allowed - cannot navigate out of configuration region

[ex:/configure]
A:admin@node-2# edit-config exclusive

(ex)[/configure]
A:admin@node-2# show

(ex)[/show]
A:admin@node-2#
```

4.2 Candidate configuration modes

To configure the router using the MD-CLI, the user must enter a configuration mode using the explicit or implicit configuration workflow.

The configuration workflow (implicit or explicit) determines if the user is restricted to the configure branch or if the user can navigate freely while in configuration mode. For more information about configuration workflows, see Implicit and explicit configuration workflows.

The configuration mode determines the interaction with other simultaneous configuration sessions. Table 15: Configuration mode overview provides an overview of the available configuration modes:

- **private configuration mode**

  See Private configuration mode for details.
- **exclusive configuration mode**

  See Exclusive configuration mode for details.
- **global configuration mode**

  See Global configuration mode for details.
- **read-only configuration mode**

  See Read-only configuration mode for details.

Table 15: Configuration mode overview

	Private configuration mode	Exclusive configuration mode	Global configuration mode	Read-only configuration mode
Candidate configuration accessed	Private candidate configuration	Global candidate configuration	Global candidate configuration	Global candidate configuration
Single vs multiple users	Multiple users can simultaneously configure their own private candidate	Only one user can configure the global candidate	Multiple users can simultaneously configure the shared global candidate	Multiple users can have simultaneous read-only access to the global candidate
Privacy	User can see own changes.	User can see own changes.	User can see changes from other global configuration sessions.	Users can see changes from

	Private configuration mode	Exclusive configuration mode	Global configuration mode	Read-only configuration mode
	Changes are not visible for read-only sessions.	Changes are visible for read-only sessions.	Changes are visible for read-only sessions.	global or exclusive configuration sessions
Commits	Own changes are committed	Own changes are committed. Commits from other configuration changes are blocked.	Changes made by all global configuration sessions are committed	Users cannot commit
Update needed?	Yes - baseline can become out-of-date when another private or global configuration session commits	No - baseline is always up-to-date. Other configuration sessions cannot commit.	Yes - baseline can become out-of-date when a private configuration session commits	No - updates are not allowed in read-only configuration mode

4.2.1 Multiple simultaneous candidate configurations

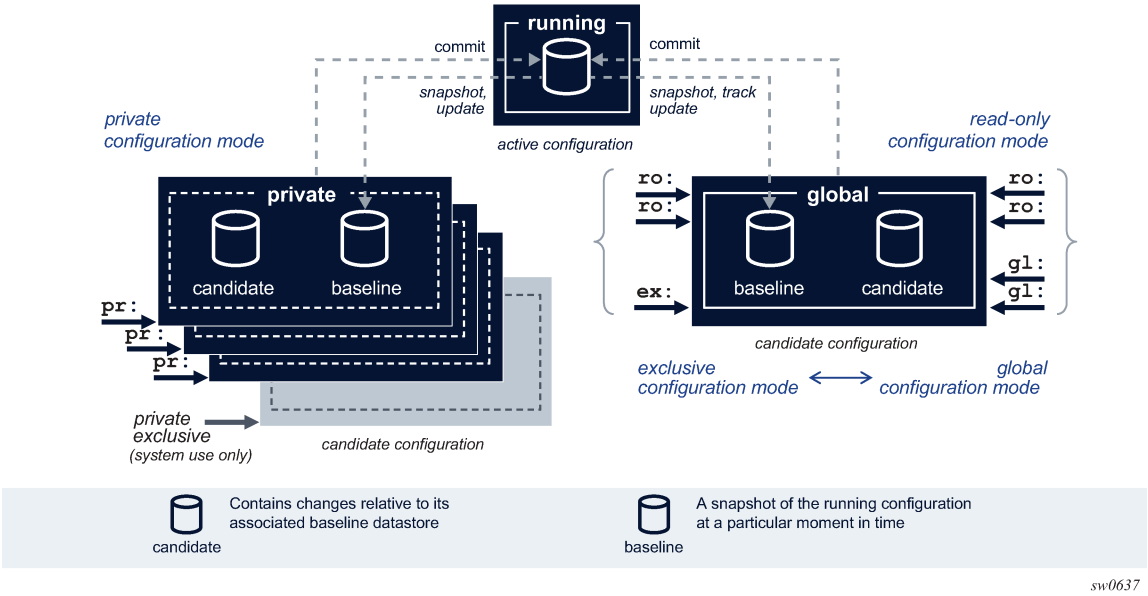
As introduced in Transactional configuration method, configuration changes are made in a candidate configuration and copied in the running configuration when the configuration changes are committed and become active.

This section describes:

- how the running configuration and a candidate configuration interact using a running datastore, a baseline datastore, and a candidate datastore
- how simultaneous configuration sessions access one or multiple candidate configurations as a function of their configuration mode

The following figure shows multiple candidate configurations.

Figure 2: Multiple candidate configurations



The running configuration is the active configuration of the router and is stored in the running datastore. There is only one running configuration in the router and therefore, only one running datastore. The running datastore is always instantiated.

The candidate configuration is a working configuration that contains changes before they are activated in the router. A candidate configuration uses two datastores:

- a baseline datastore that contains a snapshot copy of the running datastore at a specific moment in time
- a candidate datastore that contains changes relative to its associated baseline datastore

Multiple candidate configurations can exist simultaneously in the router with one of the following:

- a single global candidate configuration that is accessed by one of the following:
  - a single session in exclusive configuration mode
  - one or multiple sessions in global configuration mode
  - one or multiple sessions in read-only configuration mode

  An exclusive and global configuration session are mutually exclusive. Read-only configuration sessions can coexist with an exclusive configuration session or with one or multiple global configuration sessions.

  The global baseline datastore and global candidate datastore are always instantiated.
- up to eight private candidate configurations. A private candidate configuration is accessed by a single session in private configuration mode. The private baseline datastore and private candidate datastore are instantiated when the user enters the private configuration mode and the datastores are deleted from the router when the user exits the private configuration mode.
- one single private-exclusive candidate configuration. Only one exclusive session can be active in the router at a time: either a user-started exclusive configuration session accessing the global candidate configuration, or a system-started private-exclusive configuration session accessing a private candidate configuration, or a user-started private configuration session using NETCONF with the <lock> RPC sent

to promote it to a private-exclusive configuration session. For more information, see Private-exclusive configuration session.

When a configuration session commits its candidate configuration, the router performs the following actions:

1. verifies the running configuration has not been changed by another configuration session
2. validates the candidate configuration by verifying the logic, constraints, and completeness of the candidate configuration
3. activates the candidate configuration by sending the new candidate configuration to the corresponding applications

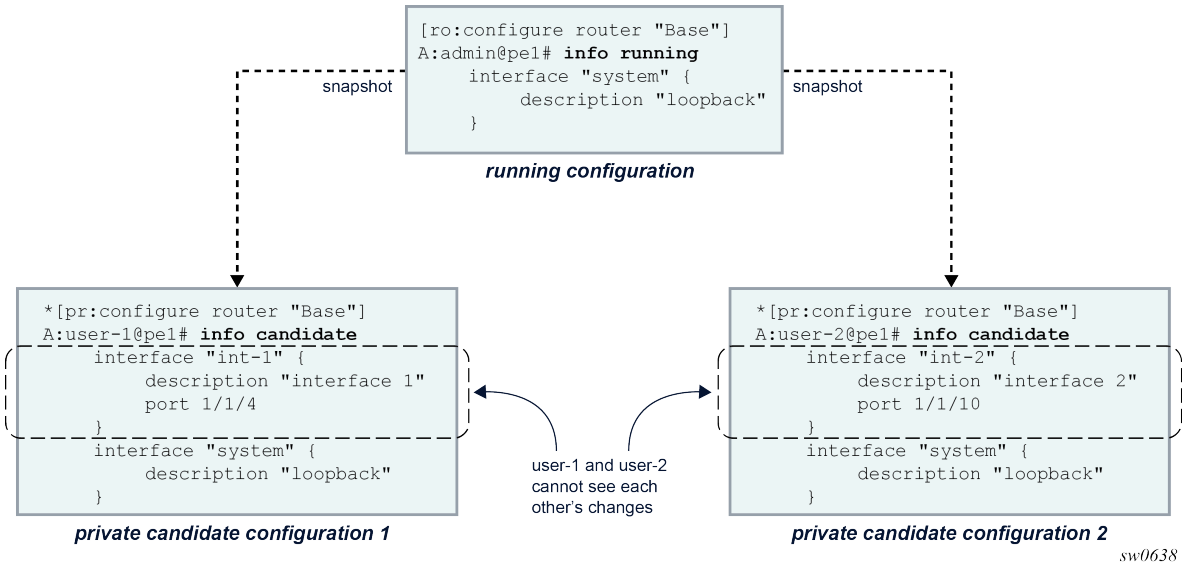
After a successful commit, the changes are copied to the running datastore, the baseline datastore contains a new copy of the running datastore, and the candidate datastore is empty.

Furthermore, when simultaneous configuration sessions access different candidate configurations:

- Multiple private configuration sessions each access their own private candidate configuration.
- One or multiple private configuration sessions each access their own private candidate configuration and one or multiple global configuration sessions all access the global candidate configuration.
- One or multiple private configuration sessions each access their own private candidate configuration and one exclusive configuration session accesses the global candidate configuration.
- One or multiple private configuration sessions each access their own private candidate configuration and one private exclusive configuration session accesses a private candidate configuration.

Each configuration session adds changes in the candidate datastore relative to the baseline associated with the candidate configuration. The baseline datastore contains a snapshot copy of the running datastore at a specific time. Therefore, multiple, simultaneous configuration sessions that are active in the router and that access different candidate configurations have their own unique view of the candidate configuration and cannot see other users' changes, as shown in the following figure.

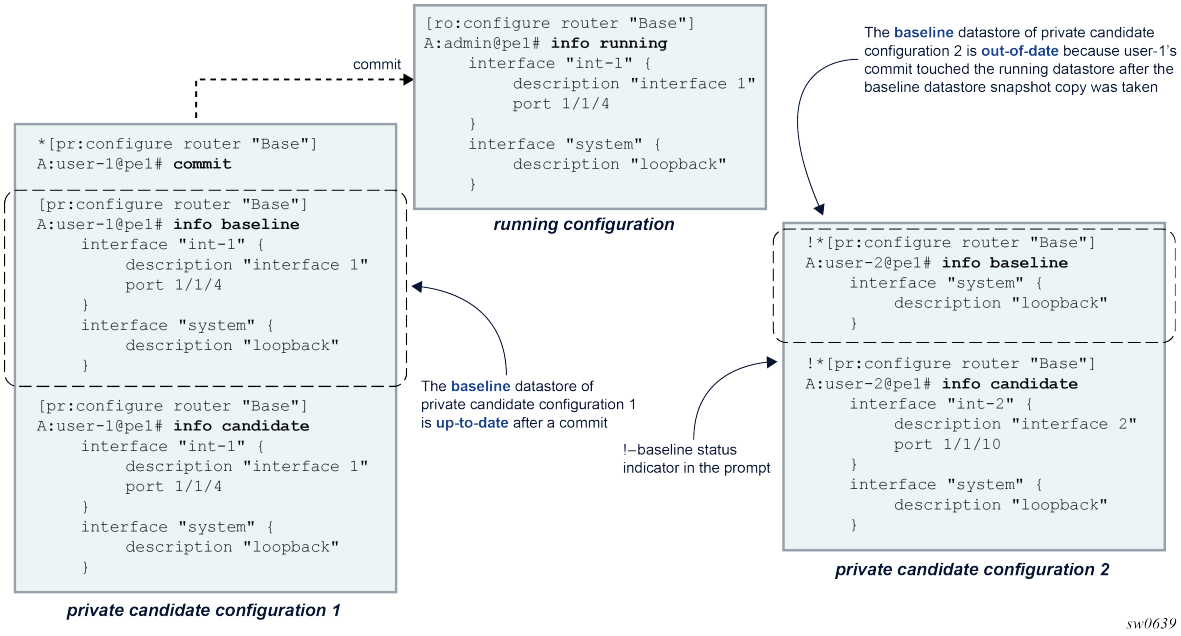
Figure 3: Simultaneous configuration sessions



Changes in a candidate configuration can only be committed when the running configuration has not been changed or touched after the baseline snapshot was taken. In other words, the baseline must be up to date to commit the changes.

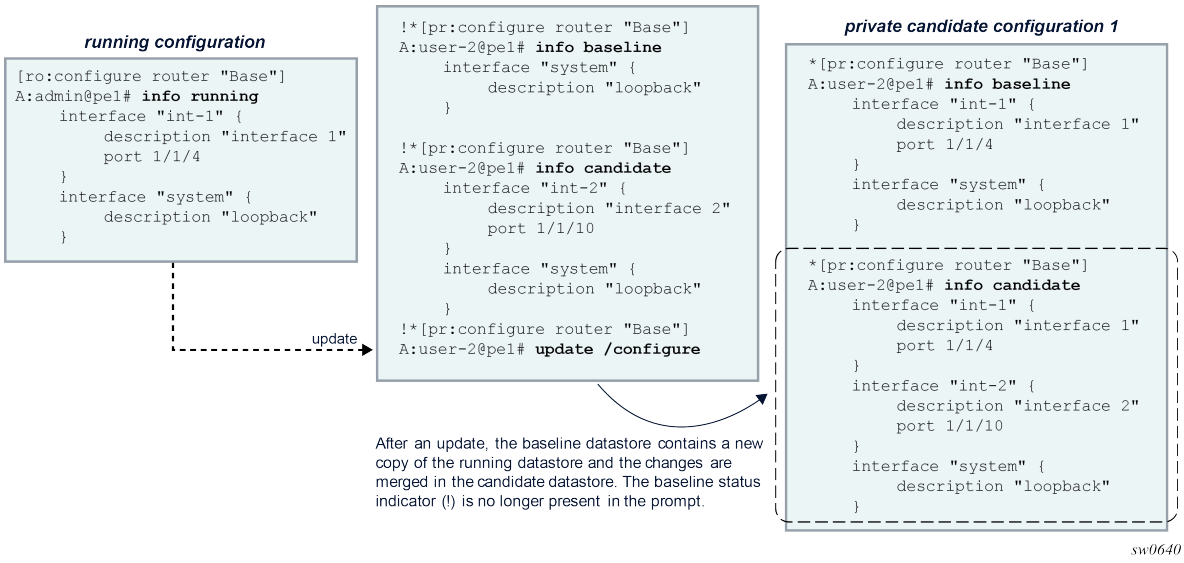
The following figure shows how the baseline datastore of user-2's candidate configuration is out-of-date after user-1 committed its changes. An exclamation mark (!) is shown in the prompt to indicate an out-of-date baseline status.

Figure 4: Simultaneous configuration sessions - baseline out-of-date



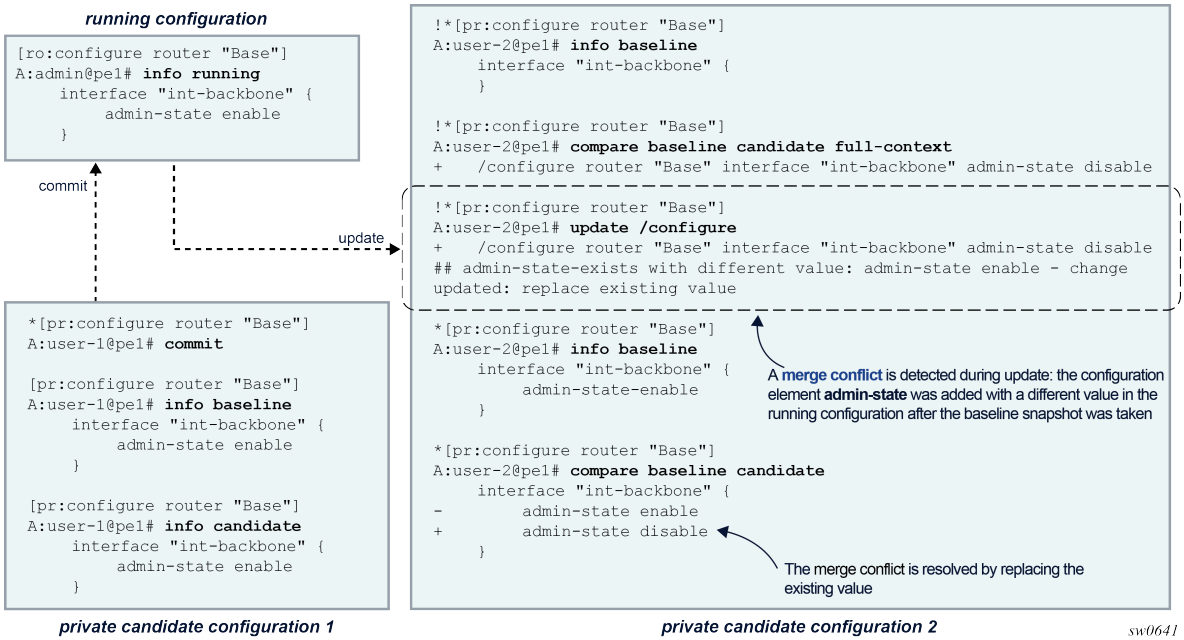
Because the baseline is out-of-date, user-2 must update its candidate configuration before committing. An update copies a new snapshot from the running datastore to the baseline datastore and merges the changes from the candidate datastore, as shown in the following figure.

Figure 5: Simultaneous configuration sessions - update



With more than one user working on the same part of the configuration, conflicts can occur when committed changes of one user's configuration session are merged into another user's candidate configuration. A merge conflict occurs when a configuration element is added, deleted, or modified in the candidate configuration and the same configuration element is also added, deleted, or modified in the running configuration after the baseline snapshot was taken. With the **update** command, the router resolves each merge conflict and installs the result in the candidate configuration, as shown in the following figure.

Figure 6: Simultaneous configuration sessions - merge conflict



When a commit operation is executed in a configuration session while the baseline is out-of-date, the router first attempts to automatically update the candidate configuration. If a merge conflict is detected, the commit operation is canceled, to allow the administrator to resolve the merge conflicts manually. The candidate configuration remains in the same state as before the commit operation.

In configuration mode, the administrator can use the following tools to check and resolve potential merge conflicts:

- **compare baseline running**

  This tool lists the changes that were made in the running datastore after a snapshot copy was stored in the baseline datastore.

- **compare baseline candidate** or **compare**

  This tool lists the candidate configuration changes.

- **update check**

  This tool performs a dry run update. The router reports all merge conflicts as if an update was performed. The candidate configuration, that is, the baseline candidate datastore, is not changed with this command.

Conflict detection and resolution is detailed in Updating the candidate configuration.

4.2.2 Private configuration mode

In private configuration mode, a private candidate configuration is reserved for editing by a single private configuration session. Each private configuration session works on its own copy of the running configuration. Only the changes made in the private configuration session are visible and can be committed. Private configuration mode can be used when multiple users are configuring simultaneously on different parts of the router configuration.

A private configuration session has the following characteristics:

- Each private configuration session accesses its own private candidate configuration. The private candidate configuration is instantiated when the user enters private configuration mode and is deleted form the router when the user exits private configuration mode.
- Changes can only be entered in its own private candidate configuration.
- Configuration changes are visible only in the private candidate configuration in which the changes are entered.
- Uncommitted changes in the private candidate configuration cannot be seen by other private, exclusive, global, or read-only configuration sessions.
- When the **commit** command is issued, only those changes entered in its own private candidate configuration are committed.
- When a private configuration session is started, a new private candidate configuration is instantiated and has no uncommitted changes.
- When a user leaves private configuration mode, uncommitted changes are discarded and the private candidate configuration is deleted. The user is prompted for confirmation to exit when uncommitted changes are present.

For simultaneous configuration sessions:

- Up to eight simultaneous private configuration sessions can coexist. Each private configuration session accesses its own private candidate configuration. Private candidate configurations can have

uncommitted changes when another private configuration session starts. A private configuration session can edit and commit its private candidate configuration while another private configuration session is active.

- An exclusive configuration session can coexist with a private configuration session. The private candidate configuration can have uncommitted changes when an exclusive configuration session starts. The exclusive session can edit and commit changes while a private configuration session is active. The private configuration session can still edit the private candidate configuration, but changes cannot be committed because the exclusive session holds a lock on the running datastore.
- Multiple global configuration sessions can coexist with a private configuration session. A global configuration session accesses the global candidate configuration. The private candidate configuration can have uncommitted changes when the global configuration session starts.
- Multiple read-only configuration sessions can coexist with a private configuration session. Read-only configuration sessions access the global candidate configuration. A read-only configuration session cannot view the changes in the private candidate configuration. The private candidate configuration can have uncommitted changes when a read-only configuration session starts.

Datastore interactions include the following characteristics:

- The private baseline datastore becomes out-of-date when another private, exclusive, global, or private exclusive configuration session commits changes to the running datastore after the private baseline snapshot was taken. An out-of-date baseline is indicated in the prompt with an exclamation mark.
- An update of the private candidate configuration is needed when its private baseline datastore is out-of-date. An update copies a new snapshot of the running datastore in the private baseline datastore and merges the changes from the private candidate datastore. Merge conflicts detected in a manual update are reported and resolved. Merge conflicts detected in an automatic update as part of a commit operation result in the cancellation of the commit operation.
- A snapshot of the running datastore is copied in the private baseline datastore:
  - at instantiation of the private candidate configuration when a user enters the private configuration mode
  - when a manual update is performed
  - after a commit, when no merge conflicts are detected during the automatic update and the updated candidate configuration is valid

When entering private configuration mode, the following messages are displayed:

```
[/]
A:admin@node-2# configure private
INFO: CLI #2070: Entering private configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit
```

Note: To display the current active configuration sessions in the router, use the command **show system management-interface configuration-sessions**.

When leaving private configuration mode, the following messages are displayed:

- without uncommitted changes in the private candidate configuration:

```
[pr:/configure]
A:admin@node-2# exit all
INFO: CLI #2074: Exiting private configuration mode
```

- with uncommitted changes present in the private candidate configuration:

```
*[pr:/configure]
A:admin@node-2# exit all
INFO: CLI #2071: Uncommitted changes are present in the candidate configuration.
Exiting private configuration mode will discard those changes.

Discard uncommitted changes? [y,n] n
INFO: CLI #2072: Exit private configuration mode canceled

*[pr:/configure]
A:admin@node-2# exit all
INFO: CLI #2071: Uncommitted changes are present in the candidate configuration.
Exiting private configuration mode will discard those changes.

Discard uncommitted changes? [y,n] y
WARNING: CLI #2073: Exiting private configuration mode - uncommitted changes are discarded
```

4.2.3 Exclusive configuration mode

In exclusive configuration mode, the global configuration is reserved for editing by a single read-write configuration session. In addition, the running datastore is locked such that no other configuration session can commit changes. Exclusive configuration mode can be used when important router configuration changes must be implemented that cannot be interrupted or delayed, and to avoid the risk of committing other users' partial completed changes.

An exclusive configuration session has the following characteristics:

- An exclusive configuration session accesses the global candidate configuration.
- Only one user can enter exclusive configuration mode at a time.
- Configuration changes in the global candidate can only be entered by the user in exclusive configuration mode.
- Configuration changes in the global candidate are visible for read-only configuration sessions.
- Changes in the global candidate configuration can only be committed by the user in exclusive configuration mode
- Uncommitted changes cannot be present in the global candidate configuration when an exclusive configuration session starts.
- Uncommitted changes are discarded from the global candidate configuration when a user leaves the exclusive configuration mode. The user is prompted for confirmation to exit when uncommitted changes are present.

For simultaneous configuration sessions:

- Multiple private configuration sessions can coexist with an exclusive configuration session. Each private configuration session accesses its own private candidate configuration. The global candidate configuration can have uncommitted changes when a private configuration session starts. A private configuration session can edit its private candidate configuration but cannot commit the changes while an exclusive configuration session is active.
- Only one exclusive configuration session can be active in the router at a time.
- An exclusive and global configuration session are mutually exclusive.

- Multiple read-only configuration sessions can coexist with an exclusive configuration session. Read-only configuration sessions access the same global candidate configuration. The global candidate configuration can have uncommitted changes when a read-only configuration session starts.

Datastore interactions include the following characteristics:

- The global baseline datastore is always up to date. Commits from other configuration sessions are blocked while an exclusive configuration session is active.
- An update of the global candidate configuration is not needed in exclusive configuration mode.

When entering exclusive configuration mode, the following messages are displayed:

- with a global configuration session active:

```
[/]
A:admin@node-2# configure exclusive
MINOR: MGMT_CORE #2052: Exclusive datastore access unavailable - model-driven interface
 editing global candidate
```

- with uncommitted changes present in the global candidate configuration:

```
[/]
A:admin@node-2# configure exclusive
MINOR: MGMT_CORE #2052: Exclusive datastore access unavailable - model-driven interface has
 uncommitted changes in global candidate
```

- with a private configuration session active:

```
[/]
A:admin@node-2# edit-config exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit
```

Note:

- MGMT_CORE #2052 is shown only when applicable.
- To display the current active configuration sessions in the router, use the command **show system management-interface configuration-sessions**.

When leaving exclusive configuration mode, the following messages are displayed:

- without uncommitted changes in the global candidate configuration:

```
[ex:/configure]
A:admin@node-2# exit all
INFO: CLI #2064: Exiting exclusive configuration mode
```

- with uncommitted changes in the global candidate configuration:

```
*[ex:/configure]
A:admin@node-2# exit all
INFO: CLI #2063: Uncommitted changes are present in the candidate configuration.
Exiting exclusive configuration mode will discard those changes.

Discard uncommitted changes? [y,n] n
INFO: CLI #2065: Exit exclusive configuration mode canceled

*[ex:/configure]
A:admin@node-2# exit all
```

```
INFO: CLI #2063: Uncommitted changes are present in the candidate configuration.
Exiting exclusive configuration mode will discard those changes.

Discard uncommitted changes? [y,n] y
WARNING: CLI #2062: Exiting exclusive configuration mode - uncommitted changes are
 discarded
```

4.2.4 Global configuration mode

In global configuration mode, the global configuration is shared with all global configuration sessions. When a user commits their changes, the changes from all users are also committed. Global configuration mode can be used when multiple users are working together on the same part of the router configuration but is generally not recommended because it can cause unintended configuration to be committed.

A global configuration session has the following characteristics:

- A global configuration session accesses the global candidate configuration.
- Multiple users can enter global configuration mode simultaneously.
- Configuration changes made by one user are visible to all other users in global or read-only configuration mode. Configuration changes in private candidate configurations are not visible.
- All changes in the global candidate configuration, from all users, are committed to the running configuration when one user commits the global candidate configuration.
- Uncommitted changes can be present in the global candidate configuration when a global configuration session starts.
- Uncommitted changes are kept in the global candidate configuration when a user leaves the global configuration mode.

For simultaneous configuration sessions:

- Multiple private configuration sessions can coexist with a global configuration session. Each private configuration session accesses its own private candidate configuration. The global candidate configuration can have uncommitted changes when a private configuration session starts.
- An exclusive and global configuration session are mutually exclusive.
- Multiple global configuration sessions can coexist. All global configuration sessions access the same global candidate configuration. The global candidate configuration can have uncommitted changes when another global configuration session starts.
- Multiple read-only configuration sessions can coexist with a global configuration session. Read-only configuration sessions access the same global candidate configuration. The global candidate configuration can have uncommitted changes when a read-only configuration session starts.

Datastore interactions include the following characteristics:

- The global baseline datastore becomes out-of-date when another private or private exclusive configuration session commits changes to the running datastore after the global baseline snapshot was taken. An out-of-date baseline is indicated in the prompt with an exclamation mark.
- An update of the global candidate configuration is needed when its global baseline datastore is out-of-date. An update copies a new snapshot of the running datastore in the global baseline datastore and merges the changes from the global candidate datastore. Merge conflicts detected in a manual update are reported and resolved. Merge conflicts detected in an automatic update as part of a commit operation result in the cancellation of the commit operation.

- The baseline datastore tracks the running datastore, that is, changes in the running datastore are automatically copied in the baseline datastore:
  - after a router reboot
  - after a successful commit
  - after a discard with an up to date global baseline
- A snapshot copy of the running datastore is copied in the global baseline datastore and tracking stops when the global candidate is touched, for example, when a configuration element has been added, deleted, or modified. A new snapshot of the running datastore is copied to the global baseline datastore when a manual update is performed.

When entering global configuration mode, the following messages are displayed:

```
[/]
A:admin@node-2# configure global
INFO: CLI #2054: Entering global configuration mode
INFO: CLI #2055: Uncommitted changes are present in the candidate configuration
INFO: CLI #2075: Other global configuration sessions are active
```

Note:

- CLI #2055 and CLI #2075 are shown only when applicable.
- To display the current active configuration sessions in the router, use the command **show system management-interface configuration-sessions**.

When leaving global configuration mode, the following messages are displayed:

```
*[gl:/configure]
A:admin@node-2# exit all
INFO: CLI #2056: Exiting global configuration mode
INFO: CLI #2057: Uncommitted changes are kept in the candidate configuration
```

Note: CLI #2057 is shown only when applicable.

4.2.5 Read-only configuration mode

In read-only configuration mode, no changes can be made to the global candidate configuration and no changes can be committed to the running configuration. Read-only configuration mode can be used when reviewing or monitoring configuration changes from other users in the global candidate configuration.

A read-only configuration session has the following characteristics:

- A read-only configuration session accesses the global candidate configuration.
- Multiple users can enter read-only configuration mode simultaneously.
- All configuration changes in the global candidate configuration are visible. Configuration changes in private candidate configurations are not visible.
- The global configuration cannot be edited and changes in the global configuration cannot be committed.
- Uncommitted changes can be present in the global candidate configuration when a read-only configuration session starts.

- Uncommitted changes are kept in the global candidate configuration when a user leaves a read-only configuration mode.

For simultaneous configuration sessions:

- Multiple private configuration sessions can coexist with a read-only configuration session. Each private configuration session accesses its own private candidate configuration. The global candidate configuration can have uncommitted changes when a private configuration session starts.
- An exclusive configuration session can coexist with a read-only configuration session. The exclusive configuration session accesses the same global candidate configuration. The global candidate configuration cannot have uncommitted changes when an exclusive configuration session starts.
- Multiple global configuration sessions can coexist with a read-only configuration session. Global configuration sessions access the same global candidate configuration. The global candidate configuration can have uncommitted changes when another global configuration session starts.
- Multiple read-only configuration sessions can coexist. Read-only configuration sessions access the same global candidate configuration. The global candidate configuration can have uncommitted changes when another read-only configuration session starts.

When entering read-only configuration mode, the following message is displayed:

```
[/]
A:admin@node-2# configure read-only
INFO: CLI #2066: Entering read-only configuration mode
```

When leaving read-only configuration mode, the following message is displayed:

```
*[ro:/configure]
A:admin@node-2# exit all
INFO: CLI #2067: Exiting read-only configuration mode
```

4.2.6 Transitioning between candidate configuration modes

Exclusive, global, and read-only configuration sessions that access the global candidate configuration can transition between these configuration modes without exiting and re-entering the configuration mode.

Transitions from and to private configuration mode are not allowed.

The following summarizes the configuration mode transitions and transitions to operational mode.

Table 16: Configuration and operational mode transitions

Configuration and operational mode transition		To				
		Global	**Exclusive**	**Read-only**	**Private**	**Operational mode**
From	**Global**	X[3]	Allowed; no other exclusive or global configuration	Allowed; uncommitted changes are kept	X	Allowed; uncommitted changes are kept

[3] Allowed, but no functional value

Configuration and operational mode transition		To				
		Global	**Exclusive**	**Read-only**	**Private**	**Operational mode**
			session can be active; uncommitted changes are kept			
	Exclusive	Allowed; uncommitted changes are discarded	X[3]	Allowed; uncommitted changes are discarded	X	Allowed; uncommitted changes are discarded
	Read-only	Allowed; no exclusive configuration session can be active; uncommitted changes are kept	Allowed; no other exclusive or global configuration session can be active; uncommitted changes are kept	X[3]	X	Allowed; uncommitted changes are kept
	Private	X	X	X	X[3]	Allowed; uncommitted changes are discarded
	Operational mode	Allowed	Allowed	Allowed	Allowed	X

Example

Transitioning from exclusive to global or read-only configuration mode causes the candidate changes to be discarded.

```
[/]
A:admin@node-2# edit-config exclusive
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

(ex)[/]
A:admin@node-2# configure router interface my-int

*(ex)[/configure router "Base" interface "my-int"]
A:admin@node-2# edit-config global
INFO: CLI #2063: Uncommitted changes are present in the candidate configuration.
Exiting exclusive configuration mode will discard those changes.

Discard uncommitted changes? [y,n] n
INFO: CLI #2065: Exit exclusive configuration mode canceled
```

```
*(ex)[/configure router "Base" interface "my-int"]
A:admin@node-2# edit-config read-only
INFO: CLI #2063: Uncommitted changes are present in the candidate configuration.
Exiting exclusive configuration mode will discard those changes.

Discard uncommitted changes? [y,n] y
WARNING: CLI #2062: Exiting exclusive configuration mode - uncommitted changes are
 discarded
INFO: CLI #2066: Entering read-only configuration mode

(ro)[/configure router "Base" interface "my-int"]
A:admin@node-2#
```

Switching from global or read-only to exclusive configuration mode is allowed when no other global or exclusive configuration session is active. Uncommitted changes in the global candidate configuration are kept.

Example

In the following example, the **admin disconnect** command is used to disconnect another active global configuration session before the current session can switch to exclusive configuration.

```
[/]
A:admin@node-2# edit-config global
INFO: CLI #2054: Entering global configuration mode
INFO: CLI #2075: Other global configuration sessions are active

(gl)[/]
A:admin@node-2# configure router interface new-int

*(gl)[/configure router "Base" interface "new-int"]
A:admin@node-2# edit-config exclusive
MINOR: MGMT_CORE #2052: Exclusive datastore access unavailable - model-driven interface
 editing global candidate

*(gl)[/configure router "Base" interface "new-int"]
A:admin@node-2# show system management-interface configuration-sessions
===============================================================================
Session ID  Region                Datastore                    Lock State
   Username                       Session Mode                 Idle Time
   Session Type                   From
-------------------------------------------------------------------------------
#22         configure             Candidate                    Unlocked
   admin                          Global                       0d 00:00:00
   MD-CLI                         135.244.144.235
 23         configure             Candidate                    Unlocked
   user-1                         Global                       0d 00:00:42
   MD-CLI                         135.244.144.235
-------------------------------------------------------------------------------
Number of sessions: 2
'#' indicates the current active session
===============================================================================

*(gl)[/configure router "Base" interface "new-int"]
A:admin@node-2#

*(gl)[/configure router "Base" interface "new-int"]
A:admin@node-2# admin disconnect session-id 23

*(gl)[/configure router "Base" interface "new-int"]
A:admin@node-2# edit-config exclusive
INFO: CLI #2056: Exiting global configuration mode
INFO: CLI #2057: Uncommitted changes are kept in the candidate configuration
```

```
INFO: CLI #2060: Entering exclusive configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

*(ex)[/configure router "Base" interface "new-int"]
A:admin@node-2#
```

4.2.7 Private-exclusive configuration session

A private-exclusive configuration session is reserved for system internal use and private candidate NETCONF sessions with an exclusive lock.

Note: Private-exclusive is not a configuration mode. Users cannot enter a private-exclusive configuration mode in the MD-CLI.

Router configuration changes are made via a private-exclusive configuration session as a result of the following scenarios:

- One of the following actions occurs when in **mixed** configuration mode:
  - any configuration is performed in the classic CLI engine
  - a gNMI configuration operation
  - a NETCONF session issuing the <lock> RPC while running in private candidate mode
- One of the following actions occurs when in **model-driven** configuration mode:
  - a gNMI configuration operation
  - a **password** command execution which causes the system to update the configuration
  - a NETCONF session issuing the <lock> RPC while running in private candidate mode

It is important to be aware that a private-exclusive configuration session can exist, as it interacts with other active configuration sessions in the following ways:

- An exclusive configuration session and a private-exclusive configuration session are mutually exclusive, as they both require a lock on the running datastore.
- The global candidate configuration and private candidate configuration can become out-of-date when changes are committed via a private-exclusive configuration session.
- Commits from global and private configuration sessions are blocked when a private-exclusive configuration session is active.
- A private-exclusive configuration session accesses its own private candidate configuration. Changes are not visible to other configuration sessions until they are committed and become active in the running configuration.

4.2.8 Restricting configuration mode sessions

It may be desirable to deny a user the ability to use specific configuration modes. For example, denying the use of exclusive configuration mode prevents the user from locking the configuration datastore, or denying the use of the global configuration mode forces the user to work in a private candidate datastore.

It is possible to use AAA to deny access to particular configuration modes, as illustrated in the following configuration example.

Example

In this example, the user *pr-user* has profile *admin-private*. Entries 3 and 4 in the local profile effectively deny users in the *admin-private* profile from entering the exclusive configuration mode in the MD-CLI.

```
[ex:/configure system security aaa local-profiles profile "admin-private"]
A:admin@node-2# info detail
## cli-session-group
    default-action permit-all
---snip---
    entry 3 {
     ## apply-groups
     ## description
        action deny
        match "edit-config exclusive"
    }
    entry 4 {
     ## apply-groups
     ## description
        action deny
        match "configure exclusive"
    }
```

```
[/]
A:pr-user@node-2# configure exclusive
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'configure'
[/]
A:pr-user@node-2# configure ?

 configure

 Configuration modes:
 global               - Enter global (shared) mode for candidate configuration.
 private              - Enter private mode for candidate configuration.
 read-only            - Enter read-only mode for candidate configuration.

          - Enter a candidate li configuration mode

[/]
A:pr-user@node-2# edit-config exclusive
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'edit-config'

[/]
A:pr-user@node-2# edit-config ?

 edit-config

 Configuration modes:
 global               - Enter global (shared) mode for candidate configuration.
 private              - Enter private mode for candidate configuration.
 read-only            - Enter read-only mode for candidate configuration.

 li                   - Enter a candidate li configuration mode
```

Example

The following additional entries to the profile deny users from entering the global configuration mode in the MD-CLI.

```
[ex:configure system security aaa local-profiles profile "admin-pr"]
```

```
A:admin@node-2# info detail

---snip---

    entry 5 {
     ## apply-groups
     ## description
        action deny
        match "configure global"
    }
    entry 6 {
     ## apply-groups
     ## description
        action deny
        match "edit-config global"
    }
```

```
[]
A:pr-user@node-2# configure ?

 configure

 Configuration modes:
 private              - Enter private mode for candidate configuration.
 read-only            - Enter read-only mode for candidate configuration.

[]
A:pr-user@node-2# edit-config ?

 edit-config

 Configuration modes:
 private              - Enter private mode for candidate configuration.
 read-only            - Enter read-only mode for candidate configuration.

 li                   - Enter a candidate li configuration mode

[]
A:pr-user@node-2# configure global
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'configure'

[]
A:pr-user@node-2# edit-config global
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'edit-config'
```

4.3 Modifying the configuration

To modify the router configuration using the MD-CLI, enter (private, exclusive, or global) configuration mode and use the available configuration commands as described in the *7705 SAR Gen 2 MD-CLI Command Reference Guide*.

To add a new configuration or make changes to the existing configuration, see Adding configuration elements. To remove a particular configuration or to return a functionality to its default condition, see Deleting configuration elements.

Note: When entering commands in the MD-CLI, whether from a configuration file or explicitly in the CLI prompt, all input after a number sign (#) is ignored.

Note: When used in a string in the MD-CLI, the following special characters must be enclosed in quotation marks ("):

- apostrophe (')
- asterisk (*)
- exclamation point (!)
- greater than (>)
- number sign (#)
- opening bracket ([)
- question mark (?)
- space (.)
- vertical bar (|)

4.4 Adding configuration elements

To add configuration statements using the MD-CLI, enter the command or parameter name with a valid value for the parameter as specified by the data type. For some parameters, it is sufficient to type the parameter name to set the parameter configuration.

The current configuration of a parameter is available via the **info detail** command, even if it is the default value or if the parameter is in an unconfigured state (indicated by ##). The display of default values allows an administrator to view the configuration, particularly in a multivendor network with different default settings. A user may choose to explicitly configure a setting that persists instead of using the default, in case the default changes.

See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for more information about configuration commands and their appropriate syntax.

4.4.1 Default values for key leafs

A leaf is an element that does not contain any other elements and has a data type, for example, a string, an integer, or an IP address.

Key leafs may have an optional default value that can be used as shorthand notation where a specified default is assumed. For example, **configure router bgp** with no instance value expands to **configure router "Base" bgp**. Default values are implemented as follows:

- Default values cannot be used in a reference.
- Multiple keys in a list can have default values.
- The first, last, or any key in a list may have a default value.
- If the first key has a default value, the other keys must be named keys.
- Default values can be used multiple times in any combination; for example, **configure router isis** expands to **configure router "Base" isis 0**, and **configure router foo isis** expands to **configure router "foo" isis 0**.
- The expansion is automatic and displayed in the command prompt context and **pwc**.

Example

```
(ex)[/]
A:admin@node-2# configure router

(ex)[/configure router "Base"]
A:admin@node-2#

(ex)[/]
A:admin@node-2# configure router isis

(ex)[/configure router "Base" isis 0]
A:admin@node-2#

(ex)[/]
A:admin@node-2# configure router ospf

(ex)[/configure router "Base" ospf 0]
A:admin@node-2# pwc
```

```
Present Working Context:
  configure
  router "Base"
  ospf 0

(ex)[/configure router "Base" ospf 0]
A:admin@node-2#
```

4.4.2 Entering integer values

Integer values can be entered in any of the following formats:

- **decimal**

  Enter an integer (whole number) without spaces; for example, 123456.

- **binary**

  Enter 0b followed by the binary value without spaces; for example, 0b1111000100100000. Negative values are not accepted.

- **hexadecimal**

  Enter 0x followed by the hexadecimal value in lowercase or uppercase without spaces; for example, 0x1E240 or 0x1e240. Negative values are not accepted.

Integer values are displayed in decimal format, unless a different output format is specified internally by the system.

Example: Integer values displayed in decimal format

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# connect-retry 0b100100101001

*[ex:/configure router "Base" bgp]
A:admin@node-2# info | match connect-retry
    connect-retry 2345

*[ex:/configure router "Base" bgp]
A:admin@node-2# connect-retry 0xd80
```

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info | match connect-retry
    connect-retry 3456

*[ex:/configure router "Base" bgp]
A:admin@node-2#
```

Example: etype parameter is a hexadecimal output value

This example shows that a decimal value can be entered, but the value is displayed in hexadecimal format.

```
*[ex:/configure filter mac-filter "fn" entry 1 match]
A:admin@node-2# etype ?

 etype <number>
 <number> - <0x600..0xffff>

   Ethernet type

*[ex:/configure filter mac-filter "fn" entry 1 match]
A:admin@node-2# etype 65535

*[ex:/configure filter mac-filter "fn" entry 1 match]
A:admin@node-2# info
    etype 0xffff
```

Note: Unions of integer and enumerated values do not support binary or hexadecimal input.

Example: Command with a union of data types

In the following example of a command with a union of data types, the **pir** command can have an integer value or it can be defined with the **max** enumerated value. If a numerical value is entered for **pir**, it must be entered as a decimal number.

```
*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# pir ?

 pir (<number> | <keyword>)
 <number>  - <1..6400000000> - kilobps
 <keyword> - max             - kilobps
 Default   - max

    Administrative PIR

*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# pir 88

*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# info
    pir 88

*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# pir 0b0010
                    ^^^^^^
MINOR: MGMT_CORE #2301: Invalid element value - 'pir' expected number
 '<1..6400000000>' (kilobps) or keyword 'max' (kilobps)
```

```
*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# info
    pir 88

*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# pir 2

*[ex:/configure qos sap-ingress "sstest" queue 8 rate]
A:admin@node-2# info
    pir 2
```

4.4.3 Configuring lists

A list is a sequence of list entries, and all keys of a list are entered on the same line as the list command. In general, the first key of a list is unnamed in the MD-CLI. All other keys are named. The name of the first key is shown in square brackets in ? help. Entering the name of the first key is optional when it is shown in brackets.

Example: Configuring a list

In the following example, **ip-address** is the first key and **port** is the second key. Entering **ip-address** in the MD-CLI is optional; entering **port** and any subsequent key name is mandatory.

```
*[ex:/configure cflowd]
A:admin@node-2# collector ?

 [ip-address] (<unicast-ipv4-address> | <global-unicast-ipv6-address>)
 <unicast-ipv4-address>        - <d.d.d.d>
 <global-unicast-ipv6-address> - (<x:x:x:x:x:x:x:x>|<x:x:x:x:x:x:d.d.d.d>)

    IP address of the remote cflowd collector host

*[ex:/configure cflowd]
A:admin@node-2# collector 10.20.30.40 ?

 port <number>
 <number> - <1..65535>

    UDP port number of the remote cflowd collector host
```

Example: Entering IP address and port number

The IP address and port number can be entered in one of the following ways.

```
*[ex:/configure cflowd]
A:admin@node-2# collector ip-address 10.10.20.30 port 7

*[ex:/configure cflowd]
A:admin@node-2# collector 10.10.20.30 port 7
```

There are some exceptions where the first key of a list is named. In these cases, the key name must be entered.

Example: Entering the key name

In the following example, the key name **index** must be entered.

```
*[ex:/configure cflowd collector 10.20.30.40 port 7 export-filter interface-list service]
A:admin@node-2# ies-interface ?

 service-name <reference>
 <reference> - <1..64 characters> - configure service ies <service-name>

    Administrative service name

*[ex:/configure cflowd collector 10.20.30.40 port 7 export-filter interface-list service]
A:admin@node-2# ies-interface service-name svc-test interface-name ?

 interface-name <reference>
 <reference> - <1..32            - configure service ies <./service-name>
               characters>         interface <interface-name>

    ies interface name

*[ex:/configure cflowd collector 10.20.30.40 port 7 export-filter interface-list service]
A:admin@node-2# ies-interface service-name svc-test interface-name int-name-test

*[ex:/configure cflowd collector 10.20.30.40 port 7 export-filter interface-list service]
A:admin@node-2# info
    ies-interface service-name "svc-test" interface-name "int-name-test" { }
```

Auto-completion does not select or complete the name of the first key if it is optional.

Example: Using an optional key name

In the following example, the key name for **ma-admin-name** is optional as indicated by the square brackets, and is not auto-completed when **Tab** is entered.

```
*[ex:/configure eth-cfm domain "dmtest"]
A:admin@node-2# association ?

 [ma-admin-name] <string>
 <string> - <1..64 characters>

    Domain association name

*[ex:/configure eth-cfm domain "dmtest"]
A:admin@node-2# association Press Tab

 <ma-admin-name>
```

If the name of the first key is optional and is not entered as part of the command, the key name can be used as the actual value of the key if it is enclosed in quotation marks.

Example: Enclosing the key name in quotation marks

```
*[ex:/configure eth-cfm domain "dmtest"]
A:admin@node-2# association "ma-admin-name"

*[ex:/configure eth-cfm domain "dmtest" association "ma-admin-name"]
A:admin@node-2# pwc
Present Working Context:
  configure
  eth-cfm
  domain "dmtest"
```

```
  association "ma-admin-name"
```

If the optional key name is entered, it can be specified as the actual value of the key with or without the quotation marks.

Example: Specifying the optional key name as the actual value of the key

```
*[ex:/configure eth-cfm domain "dmtest"]
A:admin@node-2# association ma-admin-name ma-admin-name

*[ex:/configure eth-cfm domain "dmtest" association "ma-admin-name"]
A:admin@node-2# pwc
Present Working Context:
  configure
  eth-cfm
  domain "dmtest"
  association "ma-admin-name"
```

Users can enter the wildcard asterisk (*) character, a range, or a regular expression instead of a list key name to edit multiple elements with one command. See Using expressions in commands for more information.

Example: Using the wildcard * character to enable all ports

```
[ex:/configure]
A:admin@node-2# port * admin-state enable
```

Example: Using a range to enable a range of ports

```
[ex:/configure]
A:admin@node-2# port 1/1/[1..3] admin-state enable

[ex:/configure]
A:admin@node-2# port 1/1/[1,3,5] admin-state enable
```

4.4.3.1 System-ordered lists

For system-ordered lists, list entries are automatically reordered.

Example

In the following example, the list is reordered based on the alphabetical order of the string name identifying the list instance.

```
[ex:/configure]
A:admin@node-2# eth-cfm ?

 eth-cfm

 apply-groups          - Apply a configuration group at this level
 apply-groups-exclude  - Exclude a configuration group at this level
 default-domain        + Enter the default-domain context
 domain                + Enter the domain list instance

[ex:/configure]
A:admin@node-2# eth-cfm

[ex:/configure eth-cfm]
```

```
A:admin@node-2# domain ?

 [md-admin-name] <string>
 <string> - <1..64 characters>

    Unique domain name

[ex:/configure eth-cfm]
A:admin@node-2# domain zero } domain two } domain four } domain five }

*[ex:/configure eth-cfm]
A:admin@node-2# info
    domain "five" {
    }
    domain "four" {
    }
    domain "two" {
    }
    domain "zero" {
    }
```

4.4.3.2 User-ordered lists

For user-ordered lists, new entries are appended to the end of the list.

Example: New entries appended to the list

```
[ex:/configure router "Base"]
A:admin@node-2# apply-groups grp3

*[ex:/configure router "Base"]
A:admin@node-2# apply-groups grp1

*[ex:/configure router "Base"]
A:admin@node-2# apply-groups grp9

*[ex:/configure router "Base"]
A:admin@node-2# info
    apply-groups ["grp3" "grp1" "grp9"]

*[ex:/configure router "Base"]
A:admin@node-2# apply-groups grp5

*[ex:/configure router "Base"]
A:admin@node-2# info
    apply-groups ["grp3" "grp1" "grp9" "grp5"]
```

To reorder a user-ordered list, the list can be deleted and recreated using the specified order. Alternatively, the tilde (~) character can be used to replace a list, effectively deleting and recreating the list in one step.

Example: Using the tilde (~) character to replace a user-ordered list

```
*[ex:/configure router "Base"]
A:admin@node-2# ~ apply-groups [grp1 grp3 grp5 grp8]

*[ex:/configure router "Base"]
A:admin@node-2# info
    apply-groups ["grp1" "grp3" "grp5" "grp8"]
```

It is possible to insert entries into an existing user-ordered list by using the **insert** command.

Example: List begins with two entries, named-entry “one” and named-entry “ten”

```
*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# entry-type named

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# named-entry one

*[ex:/configure policy-options policy-statement "my-ordered-list" named-entry "one"]
A:admin@node-2# back

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# named-entry ten

*[ex:/configure policy-options policy-statement "my-ordered-list" named-entry "ten"]
A:admin@node-2# back

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# info
    entry-type named
    named-entry "one" {
    }
    named-entry "ten" {
    }
```

Example: insert command is used

```
*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# insert named-entry four ?

 Global commands:
 after                - Insert a named-entry in the user-ordered list after
                        another specified named-entry
 before               - Insert a named-entry in the user-ordered list before
                        another specified named-entry
 beginning            - Insert a named-entry at the beginning of the user-
                        ordered list
 end                  - Insert a named-entry at the end of the user-ordered list

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# insert named-entry four after one

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# insert named-entry six before ten

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# info
    entry-type named
    named-entry "one" {
    }
    named-entry "four" {
    }
    named-entry "six" {
    }
    named-entry "ten" {
    }

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# insert named-entry zero beginning

*[ex:/configure policy-options policy-statement "my-ordered-list"]
```

```
A:admin@node-2# insert named-entry twenty end

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# info
    entry-type named
    named-entry "zero" {
    }
    named-entry "one" {
    }
    named-entry "four" {
    }
    named-entry "six" {
    }
    named-entry "ten" {
    }
    named-entry "twenty" {
    }
```

Example: List entries can still be deleted

```
*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# delete named-entry six

*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# info
    entry-type named
    named-entry "zero" {
    }
    named-entry "one" {
    }
    named-entry "four" {
    }
    named-entry "ten" {
    }
    named-entry "twenty" {
    }
```

Example: Adding the { keystroke

The default behavior of the **insert** command is to return immediately to the present working context. To drop into the newly-inserted entry, add the **{** keystroke, as shown in the following example.

```
*[ex:/configure policy-options policy-statement "my-ordered-list"]
A:admin@node-2# insert named-entry five after four {

*[ex:/configure policy-options policy-statement "my-ordered-list" named-entry "five"]
A:admin@node-2#
```

4.4.3.3 Special handling for lists with all key leafs

For lists in which the leafs are all keys (“key-only lists”), the creation of a single entry returns the user to the same context; that is, the MD-CLI session does not enter the context of the list member. This allows the user to enter multiple list items without the need to exit after each item.

Example

For example, **station** is a list with a single leaf that is the key. After each **station** entry, the session maintains the same context and other **station** entries can be added without applying the **back** or **exit** command.

```
*[ex:/configure router "Base" bgp monitor]
A:admin@node-2# ?

 admin-state          - Administrative state of BMP monitoring
 all-stations         - Send BMP messages to all configured stations
 apply-groups         - Apply a configuration group at this level
 route-monitoring     + Enter the route-monitoring context
 station              - Add a list entry for station

*[ex:/configure router "Base" bgp monitor]
A:admin@node-2# station stn1

*[ex:/configure router "Base" bgp monitor]
A:admin@node-2# station stn2

*[ex:/configure router "Base" bgp monitor]
A:admin@node-2# station stn3

*[ex:/configure router "Base" bgp monitor]
A:admin@node-2# info
    station "stn1" { }
    station "stn2" { }
    station "stn3" { }
```

4.4.4 Configuring leaf-lists

A leaf-list is an element that contains a sequence of values of a particular data type. Specifying a leaf-list entry in the MD-CLI is additive. New entries are added to existing entries and previous entries are not removed. If a duplicate entry is specified, the order remains.

Single or multiple leaf-list entries can be added in a single command line with the use of brackets ([]).

4.4.4.1 System-ordered leaf-lists

For leaf-lists ordered by the system, the leaf-list entries are automatically reordered, as shown in the following example.

Example: Leaf-list entries automatically reordered

```
*[ex:/configure port 1/1/2 ethernet eth-cfm mep md-admin-name "md-test" ma-admin-name "ma-
test" mep-id 8 ais]
A:admin@node-2# client-meg-level ?

 client-meg-level <value>
 client-meg-level [<value>...] - 1..7 system-ordered values separated by spaces
                                 enclosed by brackets

 <value>  - <number>
 <number> - <1..7>

    Client MEG level for AIS message generation
```

```
*[ex:/configure port 1/1/2 ethernet eth-cfm mep md-admin-name "md-test" ma-admin-name "ma-
test" mep-id 8 ais]
A:admin@node-2# client-meg-level [7 5 2 3]

*[ex:/configure port 1/1/2 ethernet eth-cfm mep md-admin-name "md-test" ma-admin-name "ma-
test" mep-id 8 ais]
A:admin@node-2# info
    client-meg-level [2 3 5 7]

*[ex:/configure port 1/1/2 ethernet eth-cfm mep md-admin-name "md-test" ma-admin-name "ma-
test" mep-id 8 ais]
A:admin@node-2# client-meg-level [4 6]

*[ex:/configure port 1/1/2 ethernet eth-cfm mep md-admin-name "md-test" ma-admin-name "ma-
test" mep-id 8 ais]
A:admin@node-2# info
    client-meg-level [2 3 4 5 6 7]
```

Example: System-ordered leaf-list reordered

The following system-ordered leaf-list is reordered based on the enumerated value of the entered keywords.

```
*[ex:/configure policy-options policy-statement "s" entry 9 from]
A:admin@node-2# family ?

 family <value>
 family [<value>...] - 1..20 system-ordered values separated by spaces enclosed
                       by brackets

 <value>   - <keyword>
 <keyword> - (ipv4|vpn-ipv4|ipv6|mcast-ipv4|vpn-ipv6|l2-vpn|mvpn-ipv4|mdt-
             safi|ms-pw|flow-ipv4|route-target|mcast-vpn-ipv4|mvpn-ipv6|
             flow-ipv6|evpn|mcast-ipv6|label-ipv4|label-ipv6|bgp-ls|mcast-
             vpn-ipv6|sr-policy-ipv4|sr-policy-ipv6)

    Address family as the match condition

*[ex:/configure policy-options policy-statement "s" entry 9 from]
A:admin@node-2# family [mcast-vpn-ipv4 bgp-ls l2-vpn]

*[ex:/configure policy-options policy-statement "s" entry 9 from]
A:admin@node-2# info
    family [l2-vpn mcast-vpn-ipv4 bgp-ls]
```

4.4.4.2 User-ordered leaf-lists

For user-ordered leaf-lists, new entries are appended to the end of the leaf-list.

Example: New entries appended to the end of the leaf-list

```
*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# from prefix-list [ plcy5 plcy1 ]

*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# info
    from {
        prefix-list ["plcy5" "plcy1"]
    }
```

```
*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# from prefix-list plcy3

*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# info
    from {
        prefix-list ["plcy5" "plcy1" "plcy3"]
    }

*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# from prefix-list plcy1

*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2# info
    from {
        prefix-list ["plcy5" "plcy1" "plcy3"]
    }

*[ex:/configure policy-options policy-statement "plcy_str" entry 29]
A:admin@node-2#
```

To reorder a user-ordered leaf-list, the user must delete and recreate the leaf-list using the specified order. See Deleting leaf-list entries and leaf-lists for more information. Alternatively, use the tilde (~) character to replace a leaf-list, effectively deleting and recreating the leaf-list in one step.

Example: Using the tilde (~) character to replace a user-ordered leaf-list

```
(ex)[/]
A:admin@node-2# configure router isis 5

*(ex)[/configure router "Base" isis 5]
A:admin@node-2# export-policy [test5 test3 test2]

*(ex)[/configure router "Base" isis 5]
A:admin@node-2# info
    export-policy ["test5" "test3" "test2"]

*(ex)[/configure router "Base" isis 5]
A:admin@node-2# ~ export-policy [test1 test2 test3 test5]

*(ex)[/configure router "Base" isis 5]
A:admin@node-2# info
    export-policy ["test1" "test2" "test3" "test5"]

*(ex)[/configure router "Base" isis 5]
A:admin@node-2#
```

4.4.5 Configuring leafs with units

If a leaf is defined by a number value and an associated unit, the user can enter the value in a different base unit than is defined. For example, if a timer is defined in seconds, it is possible to enter a value based on the number of minutes, or a combination of minutes and seconds. These dynamic units in the MD-CLI can be entered in a format that is converted into the base unit based on a conversion factor. The units for a command can be displayed using the **units** option for the **info** command.

Static units that have no conversion factor must always be entered in the base unit value; for example, a unit of packets per second, or bit errors.

Units are supported for:

- memory sizes, for example, bytes
- rates, for example, bps
- durations, for example, seconds
- dates, for example, Tue, 01 Sep 2020 15:15:35 UTC

Dynamic units can be entered as a number in one of the following ways:

- **as a value without a unit**

  The value is interpreted as the defined base unit. Decimal, binary, and hexadecimal numbers are supported.

Example: Entering transmit-interval 10 without specifying a unit

For example, **transmit-interval** has a base unit of deciseconds. Entering **transmit-interval 10**, without specifying a unit, configures the interval to 10 deciseconds.

```
[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval ?

 transmit-interval <number>
 <number> - <1..600> - deciseconds
 Default  - 10

    Transmit interval of OAMPDUs

[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 50

*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 50
```

The units for the leaf can be displayed using the **units** option of the **info** command:

- **as unique value-unit tuples**

  The units are separated by a space in any order, and the same unit cannot be used more than once. The value is interpreted as the specified unit and can only be entered as a decimal number.

Example: Acceptable formats to enter 55 deciseconds for transmit-interval

For example, there are many acceptable formats to enter 55 deciseconds for **transmit-interval**, including the following.

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 55

*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 5 seconds 5 deciseconds

*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 55

*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 5 seconds 500 milliseconds
```

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 55
```

The configured value is displayed as a positive integer in the defined base unit. Because the unit for **transmit-interval** is defined as deciseconds, the value displayed in the **info** command is in deciseconds, regardless of the format in which it was entered.

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 55
```

The input value is calculated based on the input of all input tuples and validated after **Enter** is pressed.

Example: Calculated input value

For example, entering 900 (deciseconds) for **transmit-interval** results in an error display, as 900 deciseconds is not in the element range.

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 900
                                  ^^^
MINOR: MGMT_CORE #2301: Invalid element value - 900 out of range 1..600
```

Example: Displaying valid units for a value

Entering a value followed by **Spacebar** and **Tab** displays valid units for the value, as in the following example. For a value of 900 for **transmit-interval**, the system displays valid unit possibilities, listed in alphabetical order.

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 900 Press Tab

 milliseconds  centiseconds
```

If a unit is already present in the input, it is suppressed for any further input.

Example: Units already present are suppressed

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 900 centiseconds 100 Press Tab

 milliseconds  deciseconds
```

The unit names can be singular or plural, depending on the numerical value entered. For a numerical value of 1, the unit names displayed are their singular form.

Example: Unit names are singular or plural

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# transmit-interval 1 Press Tab

 decisecond              second                  minute

...

*[ex:/configure port 1/1/1 ethernet efm-oam]
```

```
A:admin@node-2# transmit-interval 10 Press Tab

 centiseconds            deciseconds             seconds
```

Example: Displaying units for a leaf

The units for the leaf can be displayed using the **units** option of the **info** command.

```
*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info
    transmit-interval 50

*[ex:/configure port 1/1/1 ethernet efm-oam]
A:admin@node-2# info units
    transmit-interval 50 deciseconds
```

Auto-completion is supported for valid units entered after a value.

The following tables list units that have a conversion factor that allows a leaf with a specific base unit to be defined in a dynamic unit. The valid unit keywords for each unit name are also provided.

Table 17: Dynamic units for memory sizes

Unit name	Valid MD-CLI input
bytes	• bytes • byte
kilobytes	• kilobytes • kilobytes • kbytes • kbyte
megabytes	• megabytes • megabyte • mbytes • mbyte
gigabytes	• gigabytes • gigabyte • gbytes • gbyte
terabytes	• terabytes • terabyte • tbytes • tbyte

Table 18: Dynamic units for rates

Unit name	Valid MD-CLI input
bps (bits per second)	bps
kilobps (kilobits per second)	• kilobps • kbps
megabps (megabits per second)	• megabps • mbps
gigabps (gigabits per second	• gigabps • gbps
terabps (terabits per second)	• terabps • tbps
petabps (petabits per second)	• petabps • pbps
exabps (exabits per second)	• exabps • ebps
zettabps (zettabits per second)	• zettabps • zbps

Table 19: Dynamic units for duration

Unit name	Valid MD-CLI input
picoseconds	• picoseconds • picosecond • psecs • psec
nanoseconds	• nanoseconds • nanosecond • nsecs • nsec
microseconds	• microseconds • microsecond • usecs

Unit name	Valid MD-CLI input
	• usec
milliseconds	• milliseconds • millisecond • msecs • msec
centiseconds	• centiseconds • centisecond • csecs • csec
deciseconds	• deciseconds • decisecond • dsecs • dsec
seconds	• seconds • second • secs • sec
minutes	• minutes • minute • mins • min
hours	• hours • hour • hrs • hr
days	• days • day
weeks	• weeks • week • wks • wk

The following table shows the valid inputs for dates based on the time format.

Table 20: Dynamic units for dates and time

Date and time format	Valid MD-CLI input
"yyyy-mm-dd hh:mm[:ss] [TZ]" For example: "2018-06-01 13:12:59 EDT"	yyyy is RFC 3339 date-fullyear mm is RFC 3339 date-month dd is RFC 3339 date-mday hh is RFC 3339 time-hour mm is RFC 3339 time-minute, requires preceding zeros ss is RFC 3339 time-second, requires preceding zeros (optional) TZ is the time-zone name (optional) This format follows ISO 8601 and must be enclosed in quotation marks.
"[Day], dd Mon yyyy hh:mm[:ss] [TZ]" For example: "Tue, 01 Sep 2020 13:21:11 UTC"	Day is the name of the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat), dd is RFC 3339 date-mday Mon is the name of the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec) yyyy is RFC 3339 date-fullyear hh is RFC 3339 time-hour mm is RFC 3339 time-minute, requires preceding zeros ss is RFC 3339 time-second, requires preceding zeros (optional) TZ is the time-zone name (optional) This format follows RFC 1123 and must be enclosed in quotation marks.
yyyy-mm-ddThh:mm:ss[.fr][(Z|(+|-)hh:mm)] For example: 2018-05-11T13:21:11-0400 or 2018-05-11T17:21:11Z	This format follows RFC 3339 and can be enclosed in quotation marks.

4.4.6 Flexible input for MAC and IPv6 addresses

Flexible input is available for MAC and IPv6 addresses, where both uppercase and lowercase hexadecimal digits are accepted.

Example: Hexadecimal digits in an IPv6 address entered in both uppercase and lowercase

In this example, IPv6 addresses are displayed in lowercase hexadecimal digits using zero compression, according to RFC 5952, *A Recommendation for IPv6 Address Text Representation*.

```
*[ex:/configure service vprn "vprn1" dns]
A:admin@node-2# ipv6-source-address 2001:db8:aaa3::8a2e:3710:7335

*[ex:/configure service vprn "vprn1" dns]
A:admin@node-2# info
    ipv6-source-address 2001:db8:aaa3::8a2e:3710:7335
```

For MAC addresses, a dash (-) separator can be used in place of a colon (:).

Example: A dash (-) separator used in place of a colon (:)

```
*[ex:/configure qos sap-ingress "s" mac-criteria entry 5 match]
A:admin@node-2# dst-mac address aa-BB-cc-DD-eE-Ff

*[ex:/configure qos sap-ingress "s" mac-criteria entry 5 match]
A:admin@node-2# info
    dst-mac {
        address aa:bb:cc:dd:ee:ff
```

Example: Flexible input available for MAC addresses using dot (.) notation

```
*[ex:/configure filter mac-filter "str" entry 33 match]
A:admin@node-2# dst-mac address aaBB.ccDD.eEFf

*[ex:/configure filter mac-filter "str" entry 33 match]
A:admin@node-2# info
    dst-mac {
        address aa:bb:cc:dd:ee:ff
    }

*[ex:/configure filter mac-filter "str" entry 33 match]
A:admin@node-2#
```

4.4.7 Input translation

The MD-CLI supports the following input translation for UTF-8 character encoding:

- curly quotation mark to ASCII quotation mark (")
- curly apostrophe to ASCII apostrophe (')
- hyphens and dashes, including minus, en dash, em dash, and others to ASCII minus (-)

The input translation allows copy and paste functionality from word processing applications that use UTF-8 curly quotation marks, hyphens, or dashes.

4.5 Deleting configuration elements

The **delete** command removes explicit configuration and returns the element configuration to the system default state or value. If there is no defined default for an element, the element returns to an unconfigured state.

Note: Minus (**-**) can be used instead of the **delete** command.

The **delete** command can be used to delete any configuration element, such as:

- leafs
- containers
- lists
- leaf-lists

If an element has sub-elements (for example, a container with more containers and leafs), all of the sub-elements are also deleted as part of the parent deletion.

Note: If the configuration element to be removed does not exist, no warning messages are displayed.

4.5.1 Deleting leafs

The following configuration example deletes three leafs; **admin-state** and **connect-retry** return to their default values, and **description** returns to an unconfigured state.

Example

```
*[ex:/configure router "Base" bgp]
A:admin@node-2# info
    admin-state disable
    description "BGP description"
    connect-retry 65535

*[ex:/configure router "Base" bgp]
A:admin@node-2# delete admin-state

*[ex:/configure router "Base" bgp]
A:admin@node-2# delete description

*[ex:/configure router "Base" bgp]
A:admin@node-2# delete connect-retry

*[ex:/configure router "Base" bgp]
A:admin@node-2# info detail
    admin-state enable
 ## description
    connect-retry 120
    keepalive 30
    damping false
    local-preference 100
    loop-detect ignore-loop
---snip---
```

4.5.2 Deleting containers

To remove a container, the **delete** command is specified before the container name. The following examples show the deletion of a **vprn** instance from two different contexts.

Example: Removes the instance from the context configure service vprn

```
*[ex:/configure service]
A:admin@node-2# info
    vprn "vprn1" {
        description "VPRN instance 01"
        dns {
            ipv6-source-address 2001:db8:aaa3::8a2e:3710:7335
        }
        bgp {
            min-route-advertisement 50
        }
        interface "int1" {
            ipv6 {
                dhcp6 {
                    relay {
                        server ["2001:db8::" "2001:db9::" "2001:dba::" "2001:dc1::"]
                    }
                }
            }
        }
    }
*[ex:/configure service]
A:admin@node-2# delete vprn "vprn1"

[ex:/configure service]
A:admin@node-2# info detail | match vprn
 ## vprn
```

Example: Shows the deletion of the instance from the context configure

```
*[ex:/configure service]
A:admin@node-2# info
    vprn "vprn1" {
        description "VPRN instance 01"
        dns {
            ipv6-source-address 2001:db8:aaa3::8a2e:3710:7335
        }
        bgp {
            min-route-advertisement 50
        }
        interface "int1" {
            ipv6 {
                dhcp6 {
                    relay {
                        server ["2001:db8::" "2001:db9::" "2001:dba::" "2001:dc1::"]
                    }
                }
            }
        }
    }

*[ex:/configure service]
A:admin@node-2# back

*[ex:/configure]
A:admin@node-2# delete service vprn "vprn1"

[ex:/configure]
A:admin@node-2# service

[ex:/configure service]
A:admin@node-2# info detail | match vprn
```

```
 ## vprn
```

In the preceding examples, the container is returned to an unconfigured state, as indicated by the ##.

Example: Placement of the delete command

In the following example, the **timers** element is a container, which contains sub-elements that are also containers; the **lsa-generate** and **spf-wait** elements. The placement of the **delete** command determines whether the **timers** element (and all of its sub-elements) are deleted, or one of the sub-elements.

```
*[ex:/configure router "Base" ospf 0]
A:admin@node-2# info
    timers {
        lsa-generate {
            max-lsa-wait 8000
            lsa-initial-wait 10
            lsa-second-wait 1000
        }
        spf-wait {
            spf-max-wait 2000
            spf-initial-wait 50
            spf-second-wait 100
        }
    }
    area 0.0.0.0 {
    }
```

Example: delete command placed before the lsa-generate element

To delete the **lsa-generate** element and its parameters, the **delete** command is specified before the **lsa-generate** element. The **info** command shows that the **spf-wait** parameters are still configured.

```
*[ex:/configure router "Base" ospf 0]
A:admin@node-2# timers delete lsa-generate

*[ex:/configure router "Base" ospf 0]
A:admin@node-2# info
    timers {
        spf-wait {
            spf-max-wait 2000
            spf-initial-wait 50
            spf-second-wait 100
        }
    }
    area 0.0.0.0 {
    }
```

Example: delete command placed before the timers element

If the **delete** command is placed before the **timers** element, all elements within the **timers** element are also deleted.

```
*[ex:/configure router "Base" ospf 0]
A:admin@node-2# info
    timers {
        lsa-generate {
            max-lsa-wait 8000
            lsa-initial-wait 10
            lsa-second-wait 1000
        }
```

```
        spf-wait {
            spf-max-wait 2000
            spf-initial-wait 50
            spf-second-wait 100
        }
    }
    area 0.0.0.0 {
    }

*[ex:/configure router "Base" ospf 0]
A:admin@node-2# delete timers

[ex:/configure router "Base" ospf 0]
A:admin@node-2# info
    area 0.0.0.0 {
    }
```

4.5.3 Deleting list entries and lists

To remove a list entry, the delete operation is specified before the list name and the entry to be removed.

Example: Deleting a list entry

```
*[ex:/configure service]
A:admin@node-2# info | match pw-template
    pw-template "pw-1" {
    pw-template "pw-3" {
    pw-template "pw-5" {
    pw-template "pw-8" {

*[ex:/configure service]
A:admin@node-2# delete pw-template “pw-3”

*[ex:/configure service]
A:admin@node-2# info | match pw-template
    pw-template "pw-1" {
    pw-template "pw-5" {
    pw-template "pw-8" {

*[ex:/configure service]
A:admin@node-2#
```

Example: Explicit wildcard (*) deletes all members of a list

```
*[ex:/configure service]
A:admin@node-2# info | match pw-template
    pw-template "pw-1" {
    pw-template "pw-5" {
    pw-template "pw-8" {

*[ex:/configure service]
A:admin@node-2# delete pw-template *

*[ex:/configure service]
A:admin@node-2# info | match pw-template

*[ex:/configure service]
A:admin@node-2#
```

If the list is a multikey list, a combination of specific members and wildcards (*) can be used.

Example: Multikey list

In the following example, **mep** is a multikey list, where the keys are **md-admin-name**, **ma-admin-name**, and **mep-id**.

```
*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# info
    mep md-admin-name "ref1" ma-admin-name "ref2" mep-id 5 {
    }
    mep md-admin-name "ref1" ma-admin-name "ref3" mep-id 5 {
    }
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2#
```

Example: Deleting all lists with mep-id of 5

The following delete operation deletes all lists with **mep-id** of 5, regardless of the **md-admin-name** or **ma-admin-name**.

```
*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# delete mep md-admin-name * ma-admin-name * mep-id 5

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# info
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2#
```

Example: Deleting all lists where ma-admin-name is "ref3" and mep-id is 5

```
*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# info
    mep md-admin-name "ref1" ma-admin-name "ref2" mep-id 5 {
    }
    mep md-admin-name "ref1" ma-admin-name "ref3" mep-id 5 {
    }
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# delete mep md-admin-name * ma-admin-name "ref3" mep-id 5

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# info
    mep md-admin-name "ref1" ma-admin-name "ref2" mep-id 5 {
    }
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2#
```

Example: Deleting all lists where md-admin-name is "ref1"

```
*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
```

```
A:admin@node-2# info
    mep md-admin-name "ref1" ma-admin-name "ref2" mep-id 5 {
    }
    mep md-admin-name "ref1" ma-admin-name "ref3" mep-id 5 {
    }
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# delete mep md-admin-name "ref1" ma-admin-name * mep-id *

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2# info
    mep md-admin-name "ref6" ma-admin-name "ref3" mep-id 99 {
    }

*[ex:/configure service epipe "svc-name" sap 1/1/4:1 eth-cfm]
A:admin@node-2#
```

4.5.3.1 Deleting leaf-list entries and leaf-lists

To remove a leaf-list entry, the delete operation is specified before the leaf-list name and the entry to be removed.

Example: Deleting a leaf-list entry

```
*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info
    member ["profile-a" "profile-b" "profile-x"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# delete member "profile-a"

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info
    member ["profile-b" "profile-x"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2#
```

Multiple leaf-list entries can be deleted in a single command with the use of brackets. The entries do not need to be in any specific order.

Example: Deleting multiple leaf-list entries

```
*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info
    member ["profile-a" "profile-b" "profile-f" "profile-x" "profile-c"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# delete member ["profile-c" "profile-f"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info
    member ["profile-a" "profile-b" "profile-x"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2#
```

Example: Deleting all members of a leaf-list using an explicit wildcard (*)

```
*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info
    member ["profile-b" "profile-x"]

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# delete member *

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2#
```

Example: Optionally using the wildcard enclosed in brackets

```
*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# delete member [*]
```

Example: Deleting all members of a leaf-list

Deleting all members of a leaf-list sets the list to the unconfigured state (as indicated in the **info detail** display by the "##").

```
*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# delete member *

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2# info detail | match member
 ## member

*[ex:/configure system security user-params local-user user "test" console]
A:admin@node-2#
```

4.6 Copying configuration elements

4.6.1 Using copy and paste

The output from the **info** commands can be copied and pasted and used as a direct input to another MD-CLI session, or loaded from a file.

Example: Configuring identical profiles

The following example shows the output from the **info** command, displaying the following configuration for the profile of the user "guest1".

```
*[ex:/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2# info
    default-action permit-all
    entry 10 {
        action deny
        match "configure system security"
    }
    entry 20 {
```

```
        action deny
        match "configure li"
    }
    entry 30 {
        action deny
        match "show li"
    }
    entry 40 {
        action deny
        match "tools"
    }
```

The output can be copied and pasted to configure an identical profile for another user; for example, "guest2". The working context must be at the same hierarchy level, as the **info** command output is context-sensitive.

```
(ex)[/]
A:admin@node-2# configure system security aaa local-profiles profile guest2

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#
```

Copy the **info** command output and paste each line into the command line.

```
*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#     default-action permit-all

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#     entry 10 {

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 10]
A:admin@node-2#         action deny

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 10]
A:admin@node-2#         match "configure system security"

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 10]
A:admin@node-2#     }

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#     entry 20 {

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 20]
A:admin@node-2#         action deny

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 20]
A:admin@node-2#         match "configure li"

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 20]
A:admin@node-2#     }

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#     entry 30 {

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 30]
A:admin@node-2#         action deny

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 30]
A:admin@node-2#         match "show li"

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 30]
A:admin@node-2#     }
```

```
*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#     entry 40 {

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 40]
A:admin@node-2#         action deny

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 40]
A:admin@node-2#         match "tools"

*(ex)[/configure system security aaa local-profiles profile "guest2" entry 40]
A:admin@node-2#     }

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#
```

The **info** command displays the configuration changes for profile "guest2", which are identical to the configuration for profile "guest1".

```
*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2# info
    default-action permit-all
    entry 10 {
        action deny
        match "configure system security"
    }
    entry 20 {
        action deny
        match "configure li"
    }
    entry 30 {
        action deny
        match "show li"
    }
    entry 40 {
        action deny
        match "tools"
    }

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#
```

Similarly, the **info flat** command output can be copied and pasted for the user profile for "guest3".

```
*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2# info flat
    default-action permit-all
    entry 10 action deny
    entry 10 match "configure system security"
    entry 20 action deny
    entry 20 match "configure li"
    entry 30 action deny
    entry 30 match "show li"
    entry 40 action deny
    entry 40 match "tools"

*(ex)[/configure system security aaa local-profiles profile "guest2"]
A:admin@node-2#
```

```
*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2# /configure system security aaa local-profiles profile "guest3"
```

```
*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#  default-action permit-all

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 10 action deny

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 10 match "configure system security"

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 20 action deny

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 20 match "configure li"

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 30 action deny

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 30 match "show li"

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 40 action deny

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#     entry 40 match "tools"

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2# info
    default-action permit-all
    entry 10 {
        action deny
        match "configure system security"
    }
    entry 20 {
        action deny
        match "configure li"
    }
    entry 30 {
        action deny
        match "show li"
    }
    entry 40 {
        action deny
        match "tools"
    }

*(ex)[/configure system security aaa local-profiles profile "guest3"]
A:admin@node-2#
```

The output from the **info full-context** command contains the full configuration path for the configuration statements. This output can be used to reconfigure the same user profile on another router, or to rebuild the user profile if it was deleted or discarded.

Example: Deleting and re-adding the "guest1" user profile

The following example begins with a "guest1" user profile, which is subsequently deleted and re-added using the output from the **info full-context** command.

```
*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2# info full-context
```

```
      /configure system security aaa local-profiles profile "guest1" default-action
 permit-all
    /configure system security aaa local-profiles profile "guest1" entry 10 match
 "configure system security"
    /configure system security aaa local-profiles profile "guest1" entry 10 action deny
    /configure system security aaa local-profiles profile "guest1" entry 20 match
 "configure li"
    /configure system security aaa local-profiles profile "guest1" entry 30 match "show
 li"
    /configure system security aaa local-profiles profile "guest1" entry 30 action deny
    /configure system security aaa local-profiles profile "guest1" entry 40 match "tools"
    /configure system security aaa local-profiles profile "guest1" entry 40 action deny
```

The "guest1" user profile is deleted, and the **info full-context** command after the delete shows no matches for profile "guest1":

```
*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2# back

*(ex)[/configure system security aaa local-profiles]
A:admin@node-2# delete profile "guest1"

*(ex)[/configure system security aaa local-profiles]
A:admin@node-2# info full-context | match guest1

*(ex)[/configure system security aaa local-profiles]
A:admin@node-2#
```

In the next step, the original full-context output for "guest1" is copied and pasted. Because the output contains the full configuration path, the statements can be pasted from any configuration context.

```
*(ex)[/configure]
A:admin@node-2# /configure system security aaa local-profiles profile "guest1" entry 10
 match "configure system security"

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 10 action deny

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 20 match "configure li"

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 30 match "show li"

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 30 action deny

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 40 match "tools"

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#     /configure system security aaa local-profiles profile "guest1" entry
 40 action deny

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2# info
    default-action permit-all
```

```
    entry 10 {
        action deny
        match "configure system security"
    }
    entry 20 {
        action deny
        match "configure li"
    }
    entry 30 {
        action deny
        match "show li"
    }
    entry 40 {
        action deny
        match "tools"
    }

*(ex)[/configure system security aaa local-profiles profile "guest1"]
A:admin@node-2#
```

The displayed output from the **compare** command can also be used to copy and paste statements in the MD-CLI. See Viewing the uncommitted configuration changes for information about using the **compare** command.

4.6.2 Using the copy command

The **copy** command copies elements in the candidate configuration. The source and destination elements must be the same type (such as containers, lists, or leaf elements) and must be at the same configuration level.

A copied element inherits all characteristics from the source element. If a container is copied to another container element, all child elements of the container are also copied.

Example: Using the copy command

The following example shows the configuration of a BGP peer (neighbor 192.0.2.2). The **copy** command is used to deploy a new peering session (neighbor 192.0.2.200).

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info flat neighbor "192.0.2.2"
    group "mesh"
    keepalive 10
    local-preference 100
    local-as as-number 65535
    local-as private true
    add-paths ipv4 send multipaths
    add-paths ipv4 receive true
    add-paths ipv6 send multipaths
    add-paths ipv6 receive true
    import policy ["ALLOW-ALL"]
    export policy ["ALLOW-ALL"]

[ex:/configure router "Base" bgp]
A:admin@node-2# copy neighbor "192.0.2.2" to neighbor 192.0.2.200

*[ex:/configure router "Base" bgp]
A:admin@node-2# compare flat
+   neighbor "192.0.2.200" group "mesh"
+   neighbor "192.0.2.200" keepalive 10
+   neighbor "192.0.2.200" local-preference 100
```

```
+   neighbor "192.0.2.200" local-as as-number 65535
+   neighbor "192.0.2.200" local-as private true
+   neighbor "192.0.2.200" add-paths ipv4 send multipaths
+   neighbor "192.0.2.200" add-paths ipv4 receive true
+   neighbor "192.0.2.200" add-paths ipv6 send multipaths
+   neighbor "192.0.2.200" add-paths ipv6 receive true
+   neighbor "192.0.2.200" import policy ["ALLOW-ALL"]
+   neighbor "192.0.2.200" export policy ["ALLOW-ALL"]
```

Note: pwc may also be used in the source or destination to specify the present working context instead of a path.

Note: The **copy** command includes the **to** keyword, which is always evaluated as an MD-CLI keyword. Therefore, contexts that include the command **to** cannot be copied as in the above examples. It is possible to copy the element by navigating to the destination context and omitting the destination parameter (that is, defaulting to the present working context). Examples where this limitation applies include the following.

```
configure policy-options policy-statement entry to
configure policy-options policy-statement named-entry to
configure oam-pm session mpls lsp rsvp-auto to
configure router mpls lsp to
configure subscriber-management local-user-db ipoe host host-identification encap-tag-
range to
configure subscriber-management local-user-db ppp host host-identification encap-tag-
range to
```

Example: Copying configuration from one entry to another without the destination parameter

The following example copies entry "5" data to entry "10" after navigating into the context for entry "10".

```
*[ex:/configure policy-options policy-statement "s"]
A:admin@node-2# info
    entry 5 {
        to {
            level 2
        }
    }
    entry 10 {
    }

*[ex:/configure policy-options policy-statement "s"]
A:admin@node-2# entry 10

*[ex:/configure policy-options policy-statement "s" entry 10]
A:admin@node-2# copy /configure policy-options policy-statement s entry 5

*[ex:/configure policy-options policy-statement "s" entry 10]
A:admin@node-2# info
    to {
        level 2
    }
```

4.6.3 Using the rename command

The **rename** command renames a list key in the candidate configuration. Use the **rename** command instead of using the **copy** and **delete** commands for a list with a specific key.

When the **rename** command is used to rename a key for a user-ordered list, the system maintains the list order.

The **rename** command may be service-impacting, as the operations involve add and delete operations.

Note:

- Renaming a list key does not change references to the renamed key at other locations in the configuration. Changing the name of a list key without altering all references causes the MD-CLI candidate configuration to fail validation (when the **validate** or **commit** command is issued).
- Renaming a user-ordered list maintains the list order.
- **pwc** may also be used in the source to specify the present working context instead of a list key.

Example

The following example shows the BGP neighbor 192.0.2.2 renamed to neighbor 192.0.2.200. The **compare** command displays the actions performed when the configuration is committed.

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info flat neighbor "192.0.2.2"
    group "mesh"
    keepalive 10
    local-preference 100
    local-as as-number 65535
    local-as private true
    add-paths ipv4 send multipaths
    add-paths ipv4 receive true
    add-paths ipv6 send multipaths
    add-paths ipv6 receive true
    import policy ["ALLOW-ALL"]
    export policy ["ALLOW-ALL"]

[ex:/configure router "Base" bgp]
A:admin@node-2# rename neighbor "192.0.2.2" to neighbor 192.0.2.200

*[ex:/configure router "Base" bgp]
A:admin@node-2# compare flat
-   neighbor "192.0.2.2" group "mesh"
-   neighbor "192.0.2.2" keepalive 10
-   neighbor "192.0.2.2" local-preference 100
-   neighbor "192.0.2.2" local-as as-number 65535
-   neighbor "192.0.2.2" local-as private true
-   neighbor "192.0.2.2" add-paths ipv4 send multipaths
-   neighbor "192.0.2.2" add-paths ipv4 receive true
-   neighbor "192.0.2.2" add-paths ipv6 send multipaths
-   neighbor "192.0.2.2" add-paths ipv6 receive true
-   neighbor "192.0.2.2" import policy ["ALLOW-ALL"]
-   neighbor "192.0.2.2" export policy ["ALLOW-ALL"]
+   neighbor "192.0.2.200" group "mesh"
+   neighbor "192.0.2.200" keepalive 10
+   neighbor "192.0.2.200" local-preference 100
+   neighbor "192.0.2.200" local-as as-number 65535
```

```
+   neighbor "192.0.2.200" local-as private true
+   neighbor "192.0.2.200" add-paths ipv4 send multipaths
+   neighbor "192.0.2.200" add-paths ipv4 receive true
+   neighbor "192.0.2.200" add-paths ipv6 send multipaths
+   neighbor "192.0.2.200" add-paths ipv6 receive true
+   neighbor "192.0.2.200" import policy ["ALLOW-ALL"]
+   neighbor "192.0.2.200" export policy ["ALLOW-ALL"]
```

4.7 Replacing configuration elements

Use the following configuration mode command to match a pattern in candidate configuration values and replace matches with the specified pattern.

```
replace string with string
```

The default pattern match and replacement are strings. If the pattern includes a space, the pattern must be delimited by quotation marks ("). Regular expressions are delimited by apostrophes ('). For more information, see Using output modifiers.

If the replace operation introduces a syntax error, the operation is cancelled and replacements are discarded. Use the following option to specify that the replace operation continues even if there are syntax errors.

```
continue-on-error
```

Use the **depth** option to configure the depth of the replace operation from the present working context.

```
 depth
```

For example, if the present working context is **configure**, the depths of the commands are:

- **configure router** = 1
- **configure router description** = 2
- **configure router interface description** = 3

Use the following option to ignore case in the pattern match.

```
ignore-case
```

The **replace** command accepts a path parameter on the input line to replace elements starting from the specified path. If a path is not specified, the command replaces elements from the present working context.

Example: Global string replacement

In the following example, the name of route policy “RP EXPORT PEERING 64496” is replaced with “RP EXPORT PEERING 65536” globally in all contexts because the peer AS was changed from 64496 to 65536 (the peer AS change itself is not displayed for clarity).

```
[ex:/configure]
A:admin@node-2# replace “RP EXPORT PEERING 64496” with “RP EXPORT PEERING 65536”
INFO: CLI #2111: Replace - 3 replacements on 3 lines

*[ex:/configure]
A:admin@node-2# compare summary
```

```
    policy-options {
-       policy-statement “RP_EXPORT_PEERING 64496” { ... }
+       policy-statement “RP_EXPORT_PEERING 65536” {
        ---snip---
+       }
    }
    router “Base” {
        bgp {
            neighbor “10.81.195.10” {
                export {
~                   policy [“RP EXPORT PEERING 65536”]
                }
            }
            neighbor "2001:db8::31d7:0:1" {
                export {
~                   policy [“RP EXPORT PEERING 65536”]
                }
            }
        }
    }
```

Example: Number replacement

In the following example, the number 10 is replaced with 172 to change an IP address. The **continue-on-error** command is specified because the replace operation introduces a syntax error.

```
[ex:/configure router "Base" interface "system"]
A:admin@node-2# replace 10 with 172
MINOR: MGMT_CORE #2301: Invalid element value - 'ipv6-address' expected ipv6 address
 '(<x:x:x:x:x:x:x:x>|<x:x:x:x:x:x:d.d.d.d>)'
MINOR: MGMT_CORE #2510: Replace error - configure router "Base" interface "system" ipv6
 address 2001:db8::94172:0:1
INFO: CLI #2110: Replace canceled

[ex:/configure router "Base" interface "system"]
A:admin@node-2# replace 10 with 172 continue-on-error
MINOR: MGMT_CORE #2301: Invalid element value - 'ipv6-address' expected ipv6 address
 '(<x:x:x:x:x:x:x:x>|<x:x:x:x:x:x:d.d.d.d>)'
MINOR: MGMT_CORE #2510: Replace error - configure router "Base" interface "system" ipv6
 address 2001:db8::94172:0:1
INFO: CLI #2111: Replace - 1 replacement on 1 line

*[ex:/configure router "Base" interface "system"]
A:admin@node-2# compare
    ipv4 {
        primary {
-           address 10.16.193.50
+           address 172.16.193.50
        }
    }
```

Regular expression capture groups and backreferences can also be used in **replace** command patterns using special characters:

- Capture group subexpressions in the match pattern that are enclosed in parentheses specify a group to match, which is stored as a numbered variable.
- Backreferences are specified in the replacement pattern, according to the following rules:
  - The \n character sequence inserts the group matched in the subexpression, where n is a digit from 1 to 9.
  - An ampersand (&) character inserts the entire matched pattern.

– The backslash (\) and ampersand (&) characters must be escaped with a backslash (\) to be treated as a literal character.

Example: IP address renumbering with capture group and backreference

The following example replaces the prefix 172.16.100 with 10.92.55 while keeping the last number 33 to renumber the IP address.

```
[ex:/configure router "Base" interface "My Interface"]
A:admin@node-2# replace '172\.16\.100\.(.*)' with '10.92.55.\1'
INFO: CLI #2111: Replace - 1 replacement on 1 line

*[ex:/configure router "Base" interface "My Interface"]
A:admin@node-2# compare
    ipv4 {
        primary {
-           address 172.16.100.33
+           address 10.92.55.33
        }
    }
```

Example: String insertion with match and backreference

The following example shows the use of the ampersand (&) character to insert the matched pattern to change the **description** of the interface without reentering it.

```
[ex:/configure router "Base" interface "My Interface"]
A:admin@node-2# replace '.*' with 'OLD &' description
INFO: CLI #2111: Replace - 1 replacement on 1 line

*[ex:/configure router "Base" interface "My Interface"]
A:admin@node-2# compare
-   description "Important Interface"
+   description "OLD Important Interface"
```

The following usage guidelines apply to the **replace** command:

- The **replace** command may be service impacting if configuration is added or deleted.
- Nokia recommends executing the **replace** command on a candidate configuration that has no existing changes, so that only the replacements are displayed using the **compare** command.
- Multiple occurrences of the matched pattern on a single configuration line are replaced.
- Press Ctrl+C to cancel the replace operation and discard all replacements.

4.8 Commenting configuration elements

The user can add, modify, or delete comments to configuration elements using the **annotate** command. These comments are displayed directly before the element in the outputs of the **info**, **compare**, and **admin save configuration** commands.

Comments are supported in model-driven configuration mode in the **configure** and **debug** regions. They are not supported in the **bof** region.

The following usage guidelines apply to the **annotate** command:

- The input *string* is the mandatory configuration comment enclosed in quotation marks (").

- Newline separators (\n) may be entered in the comment string to display multiple comment lines.
- Comments are deleted if an empty string "" is entered. Partial deletion of a multiline comment is not supported.
- The input *path* is mandatory and is an absolute path (starting with **/configure**) or a path relative to the present working context. The key to a list entry must be specified in the path (or **pwc** to specify the present working context instead of a path).
- To be annotated, a command element must have an explicitly configured value.

Example: Using the annotate command

```
[ex:/configure]
A:admin@node-2# router bgp connect-retry 42

*[ex:/configure]
A:admin@node-2# annotate "Line one for connect retry timer \n   Line two indented" router
 bgp connect-retry

*[ex:/configure]
A:admin@node-2# compare /
    configure {
        router "Base" {
            bgp {
+               # comment: Line one for connect retry timer
+               # comment:    Line two indented
+               connect-retry 42
            }
        }
    }

*[ex:/configure]
A:admin@node-2# annotate "" /configure router bgp connect-retry

*[ex:/configure]
A:admin@node-2# compare
    router "Base" {
        bgp {
+           connect-retry 42
        }
    }
```

Comments can also be entered using the "# comment: " notation and the following usage guidelines apply:

- To enter one or more comment lines, start the line with "# comment: ". The blank space after the number sign (#) and the colon (:) is mandatory; otherwise, the comment is ignored.
- Optionally, to add multiple comment lines, use one "# comment: " with newline separators (\n).
- To delete comments, enter an empty comment "# comment: ".
- Partial deletion of a multiline comment is not supported. Entering "# comment: " deletes the entire comment.

A comment entry is ignored if a non-configuration element (such as a configuration or operational command) is entered following the comment.

Comments are not displayed in the **info** command output if any of the following options is used, even if they are used with other supported options:

- **converted** (and **model**)
- **intended**

Copying and pasting a configuration comment from an **info** command output maintains the association between the comment and element, with the assumption that the comment applies to the configuration element entered on the immediate next line.

Example: Copying and pasting a configuration comment

```
[ex:/configure router "Base" bgp]
A:admin@node-2# # comment: Line one for connect-retry

[ex:/configure router "Base" bgp]
A:admin@node-2# # comment: Line two for connect-retry

[ex:/configure router "Base" bgp]
A:admin@node-2# connect-retry 98

*[ex:/configure router "Base" bgp]
A:admin@node-2# # comment: Separate this comment \nfor deterministic-med \n note the blank
 space

*[ex:/configure router "Base" bgp]
A:admin@node-2# best-path-selection deterministic-med false

*[ex:/configure router "Base" bgp]
A:admin@node-2# info
    # comment: Line one for connect-retry
    # comment: Line two for connect-retry
    connect-retry 98
    authentication-key "qvzYGrrZtXg2eqQbniwULbem+PvCH+iycONUiGK7/g== hash2"
    ebgp-default-reject-policy {
        import false
        export false
    }
    best-path-selection {
        # comment: Separate this comment
        # comment: for deterministic-med
        # comment:  note the blank space
        deterministic-med false
        always-compare-med {
            med-value on
            strict-as false
        }
---snip---
```

```
[/]
A:admin@node-2# admin show configuration /configure router bgp
    # comment: Line one for connect-retry
    # comment: Line two for connect-retry
    connect-retry 98
    authentication-key "qvzYGrrZtXg2eqQbniwULbem+PvCH+iycONUiGK7/g== hash2"
    ebgp-default-reject-policy {
        import false
        export false
    }
    best-path-selection {
        # comment: Separate this comment
        # comment: for deterministic-med
        # comment:  note the blank space
        deterministic-med false
        always-compare-med {
            med-value on
            strict-as false
        }
```

```
    }
---snip---
```

4.9 Committing configuration changes

4.9.1 Viewing the uncommitted configuration changes

The **compare** command in the MD-CLI compares configurations and displays the difference in one output. The command can only be executed from within the **configuration** context.

The following characters are used at the beginning of the output lines, to indicate the status of the element in the configuration:

- **- (minus)**

  indicates the element is only in the first (from) configuration, displayed first
- **+ (plus)**

  indicates the element is only in the second (to) configuration, displayed second
- **(space)**

  indicates the element is unchanged
- **~ (tilde)**

  new value of the element that changed (only used in the **summary** option)
- **{...}**

  indicates deleted elements compressed to its highest container (only used in the **summary** option)

Example: Using + (plus)

```
*[ex:/configure]
A:admin@node-2# compare
    log {
+       accounting-policy 5 {
+           description "For SIO statistics"
+           collection-interval 69
+           include-system-info true
+           record service-ingress-octets
+       }
+       accounting-policy 8 {
+       }
    }
```

Note: The +/-/~ output from the **compare** command can be copied and pasted, or loaded from a file. See Using the compare outputs to copy and paste for an example.

Example: Executing the compare to running command

Executing **compare to running**, without specifying the **from** option is equivalent to **compare from running to running**, which shows no differences.

```
*[ex:/configure]
A:admin@node-2# compare to running
```

```
*[ex:/configure]
A:admin@node-2# compare to candidate
    log {
+       accounting-policy 5 {
+           description "For SIO statistics"
+           collection-interval 69
+           include-system-info true
+           record service-ingress-octets
+       }
+       accounting-policy 8 {
+       }
    }

*[ex:/configure]
A:admin@node-2# compare from running to candidate
    log {
+       accounting-policy 5 {
+           description "For SIO statistics"
+           collection-interval 69
+           include-system-info true
+           record service-ingress-octets
+       }
+       accounting-policy 8 {
+       }
    }
```

Example: Output using the flat and full-context options

```
*[ex:/configure]
A:admin@node-2# compare flat
+ log accounting-policy 5 description "For SIO statistics"
+ log accounting-policy 5 collection-interval 69
+ log accounting-policy 5 include-system-info true
+ log accounting-policy 5 record service-ingress-octets
+ log { accounting-policy 8 }

*[ex:/configure]
A:admin@node-2# compare full-context
+ /configure log accounting-policy 5 description "For SIO statistics"
+ /configure log accounting-policy 5 collection-interval 69
+ /configure log accounting-policy 5 include-system-info true
+ /configure log accounting-policy 5 record service-ingress-octets
+ /configure log { accounting-policy 8 }
```

Example: Difference between compare and compare summary commands

The following example shows the difference between the **compare** and **compare summary** commands. The **compare** command shows the deletion and addition of configuration changes, each on its own line, and the **compare summary** command shows the configuration change summarized on one line with a ~ character.

```
*[ex:/configure]
A:admin@node-2# compare
    router "Base" {
        interface "system" {
            ipv4 {
                primary {
-                   address 10.1.1.1
+                   address 10.243.5.96
                }
            }
```

```
        }
    }

*[ex:/configure]
A:admin@node-2# compare summary
    router "Base" {
        interface "system" {
            ipv4 {
                primary {
~                   address 10.243.5.96
                }
            }
        }
    }
```

Example: Required configuration changes using the compare summary command

```
*[ex:/configure policy-options policy-statement "example-policy-statement"]
A:admin@node-2# compare summary
~   insert named-entry "two" after "one"
~   insert named-entry "three" after "two"
+   named-entry "two" {
+       description "Example addition"
+       action {
+           action-type accept
+       }
+   }
```

Example: summary netconf-rpc keywords with the compare command

The following example shows the use of the **summary netconf-rpc** keywords with the **compare** command to generate an RPC that is used in a network automation environment to achieve the required configuration changes.

```
*[ex:/configure policy-options policy-statement "example-policy-statement"]
A:admin@node-2# compare netconf-rpc summary
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <policy-options>
                    <policy-statement>
                        <name>example-policy-statement</name>
                        <named-entry operation="merge" yang:insert="after" yang:key=
"[entry-name='one']">
                            <entry-name>two</entry-name>
                        </named-entry>
                        <named-entry operation="merge" yang:insert="after" yang:key=
"[entry-name='two']">
                            <entry-name>three</entry-name>
                        </named-entry>
                        <named-entry>
                            <entry-name>two</entry-name>
                            <description>Example addition</description>
                            <action>
                                <action-type>accept</action-type>
                            </action>
                        </named-entry>
                    </policy-statement>
```

```
                </policy-options>
            </configure>
        </config>
    </edit-config>
</rpc>
```

The use of the **summary netconf-rpc** options with the **compare** command provides an **edit-config** NETCONF RPC back to the YANG modeled root of the tree. With third-party YANG models, such as OpenConfig, the output is displayed when the **compare summary netconf-rpc** command is executed from within the **/configure openconfig** branch (or sub-branches) or from the absolute root (**/**). That is, third-party models are not shown when the command is executed with the **/configure** path.

Example: Using the summary netconf-rpc option

```
*[ex:/configure]
A:admin@node-2# compare summary
    system {
~       name "newtest"
    }
+   openconfig {
+       lldp {
+           config {
+               hello-timer 99
+           }
+       }
+   }

*[ex:/configure]
A:admin@node-2# compare summary netconf-rpc
<rpc message-id="3" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <system>
                    <name>newtest</name>
                </system>
            </configure>
        </config>
    </edit-config>
</rpc>

*[ex:/configure]
A:admin@node-2# compare summary netconf-rpc /configure
<rpc message-id="4" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <system>
                    <name>newtest</name>
                </system>
            </configure>
        </config>
    </edit-config>
</rpc>

*[ex:/configure]
```

```
A:admin@node-2# compare summary netconf-rpc /
<rpc message-id="5" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <system>
                    <name>newtest</name>
                </system>
            </configure>
            <lldp xmlns="http://openconfig.net/yang/lldp">
                <config>
                    <hello-timer>99</hello-timer>
                </config>
            </lldp>
        </config>
    </edit-config>
</rpc>

*[ex:/configure openconfig]
A:admin@node-2# compare summary netconf-rpc
<rpc message-id="6" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
            <lldp xmlns="http://openconfig.net/yang/lldp">
                <config>
                    <hello-timer>99</hello-timer>
                </config>
            </lldp>
        </config>
    </edit-config>
</rpc>
```

4.9.1.1 Using the compare outputs to copy and paste

In the following example, the **compare** command shows the timers that have been modified. After the **commit** command has been issued to add these to the running configuration, the **lsa-generate** container is deleted.

Note:

- The output from the **compare summary netconf-rpc** command cannot be entered into the MD-CLI using copy and paste.
- The output from UNIX **diff** of configuration commands can be entered into the MD-CLI using copy and paste when displayed with **diff --unchanged-line-format="" --old-line-format='-%L' --new-line-format='+%L' FILE1 FILE2**.

Example: Output for the compare command

```
*[ex:/configure router "Base" ospf 0 timers]
A:admin@node-2# compare
+   lsa-generate {
+       max-lsa-wait 500000
+       lsa-initial-wait 100000
+       lsa-second-wait 200000
```

```
+   }
+   spf-wait {
+       spf-max-wait 120000
+       spf-initial-wait 50000
+       spf-second-wait 60000
+   }
```

Example: compare command displays with a preceding minus

The **compare** command, using the candidate configuration as the reference, displays the same configuration statements with a preceding minus (-). These statements are used in a subsequent copy and paste function to delete some of the configuration. The minus (-) at the beginning of the configuration statement takes the place of the **delete** keyword.

```
*[ex:/configure router "Base" ospf 0 timers]
A:admin@node-2# compare from candidate to running full-context
-   /configure router "Base" ospf 0 timers lsa-generate max-lsa-wait 500000
-   /configure router "Base" ospf 0 timers lsa-generate lsa-initial-wait 100000
-   /configure router "Base" ospf 0 timers lsa-generate lsa-second-wait 200000
-   /configure router "Base" ospf 0 timers spf-wait spf-max-wait 120000
-   /configure router "Base" ospf 0 timers spf-wait spf-initial-wait 50000
-   /configure router "Base" ospf 0 timers spf-wait spf-second-wait 60000

*[ex:/configure router "Base" ospf 0 timers]
A:admin@node-2# validate

*[ex:/configure router "Base" ospf 0 timers]
A:admin@node-2# commit
```

Example: lsa-generate commands are deleted

In the next step, the **lsa-generate** commands are deleted, using a copy and paste of the first three configuration statements:

```
[ex:/configure]
A:admin@node-2# -   /configure router "Base" ospf 0 timers lsa-generate max-lsa-wait
 500000

*[ex:/configure]
A:admin@node-2# -   /configure router "Base" ospf 0 timers lsa-generate lsa-initial-wait
 100000

*[ex:/configure]
A:admin@node-2# -   /configure router "Base" ospf 0 timers lsa-generate lsa-second-wait
 200000
```

Example: Deleted lsa-generate parameters are compressed

The **compare summary** command shows that the deleted **lsa-generate** parameters are compressed to its highest container, shown with an ellipsis in braces ({}).

```
*[ex:/configure]
A:admin@node-2# compare summary
    router "Base" {
        ospf 0 {
            timers {
-               lsa-generate { ... }
            }
        }
    }
```

Example: compare summary command shows the timers container as highest deleted container

If the **timers** container is deleted, which holds both the **lsa-generate** and **spf-wait** containers, the **compare summary** command now shows the **timers** container as the highest deleted container.

```
*[ex:/configure router "Base" ospf 0]
A:admin@node-2# delete timers
*[ex:configure router "Base" ospf 0]
A:admin@node-2# compare
-   timers {
-       lsa-generate {
-           max-lsa-wait 500000
-           lsa-initial-wait 100000
-           lsa-second-wait 200000
-       }
-       spf-wait {
-           spf-max-wait 120000
-           spf-initial-wait 50000
-           spf-second-wait 60000
-       }
-   }

*[ex:/configure router "Base" ospf 0]
A:admin@node-2# compare summary
-   timers { ... }

*[ex:/configure router "Base" ospf 0]
A:admin@node-2#
```

4.9.2 Discarding configuration changes

The **discard** command in configuration mode cancels changes made to the candidate configuration without impacting the running configuration or applications. The command is available only when the MD-CLI session is in a read/write configuration mode (private, exclusive, or global configuration mode).

Example: discard operation error message

An error message is displayed when the **discard** operation is attempted from read-only configuration mode.

```
*(ro)[/configure]
A:admin@node-2# compare
    log {
+       accounting-policy 5 {
+           description "For SIO statistics"
+           collection-interval 69
+           include-system-info true
+           record service-ingress-octets
+       }
+       accounting-policy 8 {
+       }
    }

*(ro)[/configure]
A:admin@node-2# discard
MINOR: CLI #2069: Operation not allowed - currently in read-only mode
```

Uncommitted changes from a global configuration session are kept in the candidate configuration when leaving configuration mode. Uncommitted changes from an exclusive or private configuration session are discarded when leaving configuration mode and a confirmation message is displayed:

Example: Behavior of uncommitted changes

```
*(ex)[/]
A:admin@node-2# quit-config
INFO: CLI #2063: Uncommitted changes are present in the candidate configuration.
Exiting exclusive configuration mode will discard those changes.

Discard uncommitted changes? [y,n] y
WARNING: CLI #2062: Exiting exclusive configuration mode - uncommitted changes are
 discarded
INFO: CLI #2064: Exiting exclusive configuration mode
```

The **discard** command accepts a path parameter on the input line to discard candidate changes starting from the specified path. If a path is not specified, the command discards candidate changes from the present working context.

Example: Discarding candidate changes by specifying the absolute path

In the following example, from the **configure system** context, the **configure router "Base" bgp** candidate changes can be discarded by specifying the absolute path.

```
*[ex:/configure system]
A:admin@node-2# compare /configure
    log {
+       log-id 55 {
+       }
    }
+   router "Base" {
+       bgp {
+           group "group-1" {
+               admin-state enable
+               connect-retry 999
+           }
+       }
+   }
    system {
-       name "node-2"
+       name "test-name"
    }

*[ex:/configure system]
A:admin@node-2# discard /configure router "Base" bgp

*[ex:/configure system]
A:admin@node-2# compare /configure
    log {
+       log-id 55 {
+       }
    }
    system {
-       name "node-2"
+       name "test-name"
    }
```

Example: Issuing the discard command without a path parameter removes candidate changes

From the **configure system** context, issuing the **discard** command without a path parameter removes candidate changes starting from the **configure system** context (the present working context).

```
*[ex:/configure system]
A:admin@node-2# compare /configure
    log {
+       log-id 55 {
+       }
    }
    system {
-       name "node-2"
+       name "test-name"
    }

*[ex:/configure system]
A:admin@node-2# discard

*[ex:/configure system]
A:admin@node-2# compare /configure
    log {
+       log-id 55 {
+       }
    }
```

Changes made by a session that obtained an explicit lock can be discarded by disconnecting the remote session. Uncommitted changes from an exclusive configuration mode session are discarded when the session disconnects. See Viewing the status of the local datastores for information about disconnecting a session.

4.9.3 Validating the candidate configuration

The **validate** command verifies the logic, constraints, and completeness of the candidate configuration without activating any changes. A successful validation returns no errors. If the validation fails, detailed failure reasons are provided. The **validate** command can be executed from any working directory and in any configuration mode.

Note: The **validate** command does not detect configuration constraints that can only be determined when the candidate is committed, such as a syntactically correct configuration with invalid internal application dependencies, or out of resource conditions.

Example

```
*(ro)[/]
A:admin@node-2# compare
    log {
+       accounting-policy 7 {
+           description "seven"
+           collection-interval 77
+       }
    }

*(ro)[/]
A:admin@node-2# validate
```

```
*(ro)[/]
```

```
*(ex)[/]
A:admin@node-2# compare
+   eth-cfm {
+       domain "mdn" {
+           association "man" {
+               ccm-interval 10ms
+           }
+       }
+   }

*(ex)[/]
A:admin@node-2# validate
MINOR: MGMT_CORE #236: configure eth-cfm domain "mdn" level - Missing mandatory  fields
MINOR: ETH_CFM #12: configure eth-cfm domain "mdn" format - Inconsistent Value error - One
 of dns, mac, name or format must be provided
```

The **commit** command also runs validation on the configuration. Therefore, it is not necessary to execute the **validate** command as a separate step when committing the candidate configuration.

4.9.4 Updating the candidate configuration

As described in Multiple simultaneous candidate configurations, a candidate configuration uses two datastores:

- a baseline datastore that contains a snapshot copy of the running configuration at a specific time
- a candidate datastore that contains changes relative to its associated baseline datastore

For a private candidate configuration, access by MD-CLI sessions in private configuration mode, a snapshot of the running configuration is copied in the private baseline datastore:

- when a private candidate configuration is instantiated, when a user enters the private configuration mode
- when a manual update is performed
- after a commit, when no merge conflicts are detected during the automatic update and the updated candidate configuration is valid

For the global candidate configuration, accessed by MD-CLI sessions in global and exclusive configuration mode, a tracking mechanism exists.

- The baseline datastore tracks the running datastore, that is, changes in the running datastore are automatically copied in the baseline datastore:
  - after a router reboot
  - after a successful commit
  - after a discard with an up-to-date global baseline
- Tracking stops and a snapshot of the running datastore is copied in the global baseline datastore when the global candidate is touched (for example, a configuration element is added, deleted, or modified). A new snapshot of the running datastore is copied in the global baseline datastore when a manual update is performed.

With two simultaneous active configuration sessions that access different candidate configurations, a commit from one configuration session changes the running configuration and causes the candidate configuration of the other session to be out of date and must be updated.

To update a candidate configuration, the following tasks are performed:

1. A new snapshot of the running configuration is copied in the baseline datastore.
2. The candidate configuration changes are merged in the new baseline:
   a. The changes in the candidate datastore are applied to the new baseline datastore.
   b. Merge conflicts are detected and resolved. A merge conflict occurs when a configuration element is added, deleted, or modified in the candidate configuration and the same configuration element was also added, deleted, or modified in the running configuration after the baseline snapshot was taken.
   c. The resulting changes are stored in the candidate datastore as new changes relative to the updated baseline.

An update can be performed manually with the **update** command. The update must be executed at the configuration root (**/configure**). Merge conflicts are reported and resolved according to the conflict resolution rules. The **update** command does not provide output when no conflicts are detected.

Example: Merge conflict reported in an update

```
+   /configure router "Base" interface "int-1" ipv4 primary address 10.2.3.4
##  address - exists with different value: address 10.1.2.3 - change updated: replace
 existing value
```

The first line lists the candidate configuration change that caused the merge conflict, in this case, adding an interface IPv4 address.

The second line describes the merge conflict and starts with a double hash (##) followed by the description:

- A merge conflict is detected for the configuration element **address**.
- The address already exists in the running configuration, but has a different value.
- The candidate configuration change as shown on the first line is updated; instead of adding an interface address, the interface address is replaced.

An update is automatically started when the candidate configuration is committed. The commit is canceled when merge conflicts are detected to give the administrator the opportunity to resolve the conflicts before committing again. The update, in this case, is not executed, the candidate configuration is unchanged, and the baseline datastore is not updated.

The **update check** command performs a dry-run update of the candidate configuration. Merge conflicts are reported the same way as for the **update** command, but the update is not executed. The **update check** command must be executed at the configuration root (**/configure**) or it can be executed in any configure branch descendant as **update check /configure**.

4.9.4.1 Example update scenario with merge conflicts

The private candidate configuration of user-1 is out of date. The running configuration has interface **backbone-1** configured. The private baseline datastore does not have the interface configured. The interface **backbone-1** configured by user-1 has a different address in its candidate configuration.

Example: Update scenario with merge conflicts

```
!*[pr:/configure router "Base"]
A:user-1@node-2# info running
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.2.2
                prefix-length 24
            }
        }
    }

!*[pr:/configure router "Base"]
A:user-1@node-2# info baseline

!*[pr:/configure router "Base"]
A:user-1@node-2# info
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.1.1
                prefix-length 24
            }
        }
    }
```

The following is a list of changes entered in the private candidate configuration of user-1.

```
!*[pr:/configure router "Base"]
A:user-1@node-2# compare baseline candidate full-context summary
+   /configure router "Base" interface "backbone-1" { }
+   /configure router "Base" interface "backbone-1" { ipv4 primary }
+   /configure router "Base" interface "backbone-1" ipv4 primary address 192.168.1.1
+   /configure router "Base" interface "backbone-1" ipv4 primary prefix-length 24
```

A **commit** command starts an automatic update. Because merge conflicts are detected, the **commit** is canceled:

```
!*[pr:/configure router "Base"]
A:user-1@node-2# commit
MINOR: MGMT_CORE #2703: Commit canceled - conflicts detected, use update
```

A dry-run update detects the merge conflicts without executing the update. Each configuration element that is changed in both the candidate configuration and the running configuration after the last baseline snapshot was taken results in a conflict and is reported.

```
!*[pr:/configure]
A:user-1@node-2# update check
+   /configure router "Base" { interface "backbone-1" }
##  interface "backbone-1" { } - already exists - change removed

+   /configure router "Base" { interface "backbone-1" ipv4 primary }
##  primary { } - already exists - change removed

+   /configure router "Base" interface "backbone-1" ipv4 primary address 192.168.1.1
##  address - exists with different value: address 192.168.2.2 - change updated: replace
 existing value

+   /configure router "Base" interface "backbone-1" ipv4 primary prefix-length 24
```

```
##  prefix-length - exists with same value - change removed
```

After verifying that the merge conflict resolution is acceptable, the update can be executed. The reporting is the same as for a dry-run update.

```
!*[pr:/configure]
A:user-1@node-2# update
+   /configure router "Base" { interface "backbone-1" }
##  interface "backbone-1" { } - already exists - change removed

+   /configure router "Base" { interface "backbone-1" ipv4 primary }
##  primary { } - already exists - change removed

+   /configure router "Base" interface "backbone-1" ipv4 primary address 192.168.1.1
##  address - exists with different value: address 192.168.2.2 - change updated: replace
 existing value

+   /configure router "Base" interface "backbone-1" ipv4 primary prefix-length 24
##  prefix-length - exists with same value - change removed
```

The candidate configuration is now updated: the baseline datastore equals the running datastore and the candidate datastore contains the updated list of changes as described in the update report.

```
*[pr:/configure router "Base"]
A:user-1@node-2# compare baseline candidate
    interface "backbone-1" {
        ipv4 {
            primary {
-               address 192.168.2.2
+               address 192.168.1.1
            }
        }
    }

*[pr:/configure router "Base"]
A:user-1@node-2# info
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.1.1
                prefix-length 24
            }
        }
    }

*[pr:/configure router "Base"]
A:user-1@node-2# info baseline
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.2.2
                prefix-length 24
            }
        }
    }

*[pr:/configure router "Base"]
A:user-1@node-2# info running
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.2.2
```

```
                prefix-length 24
            }
        }
    }
```

4.9.4.2 Example update scenario without merge conflicts

The private candidate configuration of user-1 is out-of-date. The running configuration has interface **backbone-1** configured. The private baseline datastore does not have the interface configured. The interface **backbone-2** is configured by user-1.

Example: Update scenario without merge conflicts

```
!*[pr:/configure router "Base"]
A:user-1@node-2# info running
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.1.1
                prefix-length 24
            }
        }
    }

!*[pr:/configure router "Base"]
A:user-1@node-2# info baseline

!*[pr:/configure router "Base"]
A:user-1@node-2# info
    interface "backbone-2" {
        ipv4 {
            primary {
                address 192.168.2.2
                prefix-length 24
            }
        }
    }
```

The following shows the list of changes entered in the private candidate configuration of user-1.

```
!*[pr:/configure]
A:user-1@node-2# compare baseline candidate full-context summary
+   /configure router "Base" interface "backbone-2" ipv4 primary
+   /configure router "Base" interface "backbone-2" ipv4 primary address 192.168.2.2
+   /configure router "Base" interface "backbone-2" ipv4 primary prefix-length 24
```

A dry-run update detects merge conflicts without executing the update. There are no conflicts detected in this case.

```
!*[pr:/configure]
A:user-1@node-2# update check
```

A commit operation starts an automatic update. Without merge conflicts, the commit succeeds.

```
!*[pr:/configure]
A:user-1@node-2# commit

[pr:/configure]
```

```
A:user-1@node-2#
```

After a commit operation, the candidate configuration is updated; the baseline datastore equals the running datastore and the candidate datastore is empty.

```
[pr:/configure]
A:user-1@node-2# compare baseline candidate

[pr:/configure]
A:user-1@node-2# compare baseline running

[pr:/configure router "Base"]
A:user-1@node-2# info
    interface "backbone-1" {
        ipv4 {
            primary {
                address 192.168.1.1
                prefix-length 24
            }
        }
    }
    interface "backbone-2" {
        ipv4 {
            primary {
                address 192.168.2.2
                prefix-length 24
            }
        }
    }
```

4.9.5 Committing the candidate configuration

The **commit** command can be executed from any hierarchy level within any configuration branch.

Note: The **confirmed** option of the **commit** command is only available for the **configure** configuration region.

When a commit operation is initiated while the baseline is out-of-date, the router first attempts to update the candidate configuration. If a merge conflict is detected, the commit operation is canceled to allow the administrator to resolve the merge conflicts manually.

Example: Canceled commit operation

```
!*[pr:/configure]
A:admin@node-2# commit
MINOR: MGMT_CORE #2703: Commit canceled - conflicts detected, use update

!*[pr:/configure]
A:admin@node-2#
```

The update is executed and the commit operation proceeds when no merge conflict is detected. See Updating the candidate configuration for information about the update process.

Validation is subsequently performed on the candidate configuration.

With a successful validation, the changes are copied to the running configuration, which becomes the current, operational router configuration. The candidate configuration is reset to its initial state; an empty candidate datastore and an up-to-date baseline.

If the commit operation fails, an automatic rollback occurs, which returns the running state to the state before the commit was applied. An automatic rollback does not use a rollback checkpoint file, so is not dependent on persistency to be enabled. Instead, a list of changes is kept in memory until the automatic rollback is completed. The uncommitted changes remain in the candidate configuration.

For information about the commit history, see the *7705 SAR Gen 2 System Management Guide*, "Commit History" section.

Example: Optional comment added to the commit operation

```
[pr:/configure system time]
A:admin@node-2# info
    prefer-local-time true

[pr:/configure system time]
A:admin@node-2# prefer-local-time false

*[pr:/configure system time]
A:admin@node-2# commit comment "Revert to use UTC time."

[pr:/configure system time]
A:admin@node-2# info
    prefer-local-time false
```

4.9.5.1 Using the commit confirmed command

Executing the **commit** command with no options performs the operation immediately. the confirmed option can be used to activate configuration changes without making them persistent, to give the user time to verify that the configuration is working as intended. By default, the **commit confirmed** command executes the **commit** operation with an automatic rollback of 10 minutes. Within this time, an explicit confirmation (**commit confirmed accept**) must be issued for the changes to become persistent. Other configuration commands issued during this time interval are blocked.

Note: Commit confirmed is not supported in the Boot Option File (BOF), debug, or Lawful Intercept (LI) configuration regions.

While the **commit confirmed** timer is running, the remaining time before an automatic rollback is shown before each prompt of all active MD-CLI sessions.

Example: commit confirmed timer

```
*[ex:/configure log accounting-policy 5]
A:admin@node-2# commit confirmed

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 59 seconds
[ex:/configure log accounting-policy 5]
A:admin@node-2#

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 47 seconds
[ex:/configure log accounting-policy 5]
A:admin@node-2# pwc
Present Working Context:
  configure
  log
  accounting-policy 5

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 45 seconds
[ex:/configure log accounting-policy 5]
```

```
A:admin@node-2# back

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 41 seconds
[ex:/configure log]
A:admin@node-2# accounting-policy 9
MINOR: MGMT_CORE #2604: Commit confirmed in progress - changes to the candidate
 configuration are not allowed

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 38 seconds
[ex:/configure log]
A:admin@node-2#

INFO: CLI #2090: Commit confirmed - automatic rollback in 8 minutes 44 seconds
[ex:/configure log]
A:admin@node-2#
```

If the initial **commit** fails, the **commit confirmed** operation is canceled and no timer is started.

Example: When the initial commit fails

```
*[ex:/configure log accounting-policy 5]
A:admin@node-2# collection-interval 3

*[ex:/configure log accounting-policy 5]
A:admin@node-2# commit confirmed
MINOR: LOG #12: configure log accounting-policy 5 collection-interval - Inconsistent Value
 error - Minimum value is 5 minutes for this record type.

*[ex:/configure log accounting-policy 5]
A:admin@node-2#
```

The **timeout** option for the **commit confirmed** operation can override the default value of 10 minutes. While a **commit confirmed** timer is running, a subsequent **commit confirmed** or **commit confirmed** operation with a timeout option restarts the timer.

Example: Using the timeout option

```
*[ex:/configure log accounting-policy 5]
A:admin@node-2# commit confirmed

INFO: CLI #2090: Commit confirmed - automatic rollback in 10 minutes
[ex:/configure log accounting-policy 5]
A:admin@node-2# commit confirmed 33

INFO: CLI #2090: Commit confirmed - automatic rollback in 33 minutes
[ex:/configure log accounting-policy 5]
A:admin@node-2#
```

After the **commit confirmed** operation is underway, the timer starts. A **commit confirmed cancel** command terminates an ongoing confirmed commit and immediately performs an automatic rollback to the previous state before the initial **commit confirmed** command was issued.

If the **commit confirmed accept** command is not issued within the specified timeout period after a successful commit, all changes are automatically discarded from the running configuration. If the configuration session from which the commit confirmed was initiated is still active, the candidate configuration maintains all uncommitted configuration changes.

Note: The **admin system management-interface commit confirmed** commands accept or cancel a commit confirmed that is in progress and that was started by another configuration session.

4.9.5.1.1 Non-persistent operation

The **commit confirmed** and **commit confirmed accept** or **commit confirmed cancel** commands must be executed from the same MD-CLI configuration session. Commit commands executed from another configuration session while the **commit confirmed** timer is running generate an error.

Leaving the configuration mode or logging out from the MD-CLI session cancels the ongoing **commit confirmed** and starts an automatic rollback. The user must acknowledge the request to exit configuration mode or logout.

Example

```
*[ex:/configure]
A:admin@node-2# commit confirmed

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 59 seconds
[ex:/configure]
A:admin@node-2# exit all
INFO: CLI #2095: Commit confirmed in progress - exiting configuration mode will cancel the
 commit confirmed and start configuration rollback
Cancel commit confirmed and rollback immediately? [y,n] n
INFO: CLI #2065: Exit exclusive configuration mode canceled

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 45 seconds
[ex:/configure]
A:admin@node-2# logout
INFO: CLI #2095: Commit confirmed in progress - logout will cancel the commit confirmed
 and start configuration rollback
Cancel commit confirmed and rollback immediately? [y,n] n
INFO: CLI #2100: Logout canceled

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 37 seconds
[ex:/configure]
A:admin@node-2#
```

4.9.5.1.2 Persistent identifier

Note: In private configuration mode, **commit confirmed** with a persistent identifier cannot be used. Instead, use the non-persistent **commit confirmed** command.

A persistence identifier can be specified with the initial **commit confirmed** command. A **commit confirmed accept** or **cancel** command can then be executed from the same or a different MD-CLI configuration or NETCONF session, from where the **commit confirmed persist-id** command was initiated. The persistence identifier must then be included with the subsequent **commit confirmed** commands. The persistence identifier is a user-defined string of up to 255 characters or an empty string ("").

Example

```
*[ex:/configure]
A:admin@node-2# commit confirmed persist-id my-commit
```

```
INFO: CLI #2090: Commit confirmed - automatic rollback in 10 minutes
[ex:/configure]
A:admin@node-2# commit confirmed cancel
MINOR: MGMT_CORE #2603: Commit confirmed - persist-id expected

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 53 seconds
[ex:/configure]
A:admin@node-2# commit confirmed accept
MINOR: MGMT_CORE #2603: Commit confirmed - persist-id expected

INFO: CLI #2090: Commit confirmed - automatic rollback in 9 minutes 45 seconds
[ex:/configure]
A:admin@node-2# commit confirmed cancel persist-id my-commit

*[ex:/configure]
A:admin@node-2#
```

4.10 Saving changes

The running configuration can be saved to a local or remote file location with the **admin save** [**url**] **location** command, where *location* is a character string specifying the local or remote location where the configuration is to be saved.

To make the running configuration persistent, the configuration should be saved to the startup configuration location specified in the Boot Options File (BOF) as **bof configuration primary-location**. This is achieved with the **admin save** command without specifying a location.

The MD-CLI has an implicit persistency option linked to the **commit** command: the **auto-config-save** command in **configure system management-interface cli md-cli**. When candidate configuration changes are successfully committed, the configuration is automatically saved if **auto-config-save** is set to **true**.

The configuration is saved automatically after every successful commit when automatic save is enabled via the MD-CLI, NETCONF, or gRPC **auto-config-save** commands. Nokia recommends that the **auto-config-save** values are configured the same for all model-driven interfaces.

The configuration is also saved if a user issues the **admin save** command, regardless of the **auto-config-save** settings.

When **auto-config-save** is set to **false**, the **admin save** command must be issued to make the configuration persistent.

Note: Configuration changes made with CLI scripts executed from CRON or EHS must be saved using the **admin save** command, regardless of the **auto-config-save** settings. Configuration changes made with Python applications executed from CRON or EHS are saved according to the NETCONF **auto-config-save** setting.

Example: Automatic save disabled

```
[ex:/configure system management-interface cli md-cli]
A:admin@node-2# info detail auto-config-save
    auto-config-save false
```

4.11 Rolling back the configuration from a saved configuration file

The **rollback** command loads a previously saved configuration file to the candidate configuration. Loading the file does not automatically initiate a **commit** command, so that the candidate can be examined before committing. The **rollback** command is the equivalent of a **load full-replace** with a saved configuration file, and is specified by a saved configuration number or a commit history identifier. The keyword **startup** and saved configuration number 0 correspond to the last saved configuration file (`config.cfg`). This is also the default when the identifier is not specified with the **rollback** command.

The **rollback** command is available only in model-driven management interface configuration mode and can only be executed from the root of the configuration branch.

Configuration files loaded with the **rollback commit-id** command are identified with a number that corresponds to the commit history identifier and location specified by the **bof configuration primary-location** command in the active CPM Boot Option File (BOF).

By default, 50 configuration files are saved. The **configuration-backups** command can be used to save a different number of configuration files.

```
[ex:/configure system management-interface configuration-save]
A:admin@node-2# info detail
 ## apply-groups
 ## apply-groups-exclude
    configuration-backups 50
    incremental-saves true

[ex:/configure system management-interface configuration-save]
A:admin@node-2# configuration-backups ?

 configuration-backups <number>
 <number> - <1..200>
 Default  - 50

    Maximum number of configuration versions maintained
```

Example: Displaying the name of the saved configuration file

The **admin show configuration bof** command shows the name of the saved configuration file as `config.cfg`.

```
[/]
A:admin@node-2# admin show configuration bof | match primary-location
        primary-location "cf3:\config.cfg"
```

Example: Executing the rollback command to the configuration saved before the last one

```
[ex:/configure]
A:admin@node-2# rollback 1
Loaded 326 lines in 0.5 seconds from file "cf3:\config.cfg.1"
```

Example: Executing the rollback command to the configuration saved three times before the last one

```
[ex:/configure]
A:admin@node-2# rollback 3
Loaded 306 lines in 0.4 seconds from file "cf3:\config.cfg.3"
```

Example: Executing the rollback command with a commit history identifier

In the MD-CLI, the **rollback commit-id** command references the file in the location of the commit history identifier, in this example, identifier 3.

```
[ex:/configure]
A:admin@node-2# rollback commit-id Press Tab

 <commit-id>
 4
  Committed 2024-01-08T15:27:42.9-05:00 by admin (NETCONF) from 10.1.236.68
  Comment   "Third commit with NETCONF."
  Location  "cf3:\config.cfg"
 3
   Committed 2024-01-08T15:25:51.5-05:00 by admin (MD-CLI) from 10.1.145.205
   Comment   "Second commit with the MD-CLI."
   Location  "cf3:\config.cfg.1"
 2
   Committed 2024-01-08T15:25:45.1-05:00 by admin (MD-CLI) from 10.1.145.205
   Comment   "First commit with the MD-CLI."
   Location  "cf3:\config.cfg.2"
 1
   Committed 2024-01-08T08:59:34.8-05:00 by system (MD-CLI) from Console
   Log       "System booted version 23.10.R1."
   Location  "Configuration is not saved to startup."

[ex:/configure]
A:admin@node-2# rollback commit-id 3
Executed 386 lines in 0.4 seconds from file cf3:\config.cfg.1
```

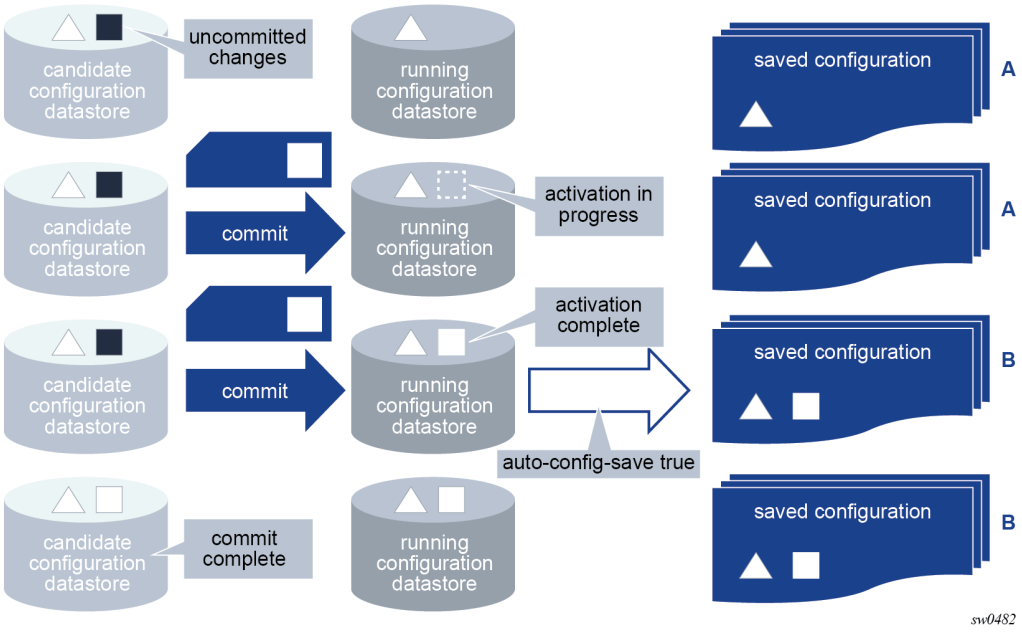
The **startup** option of the **rollback** command loads the contents of the saved configuration file and not the version of the startup file that was loaded when the system booted.

When **auto-config-save** is set to **true**, the **rollback** command without an identifier is the equivalent of executing the **discard** command for the current candidate configuration changes.

The following figures show the relationship between the candidate and running configurations, the **commit** command, the setting of the **auto-config-save** parameter, and the saved configuration files.

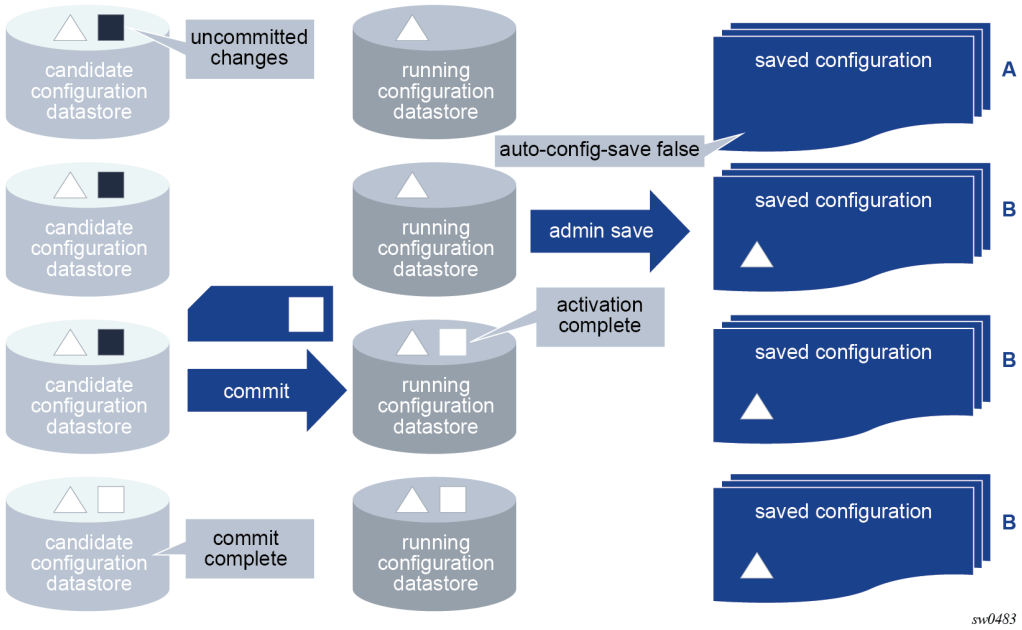
In the following figure, **auto-config-save** is set to **true**. With a successful commit, a commit history entry is created and the configuration file is saved.

Figure 7: Successful commit with auto-config-save true



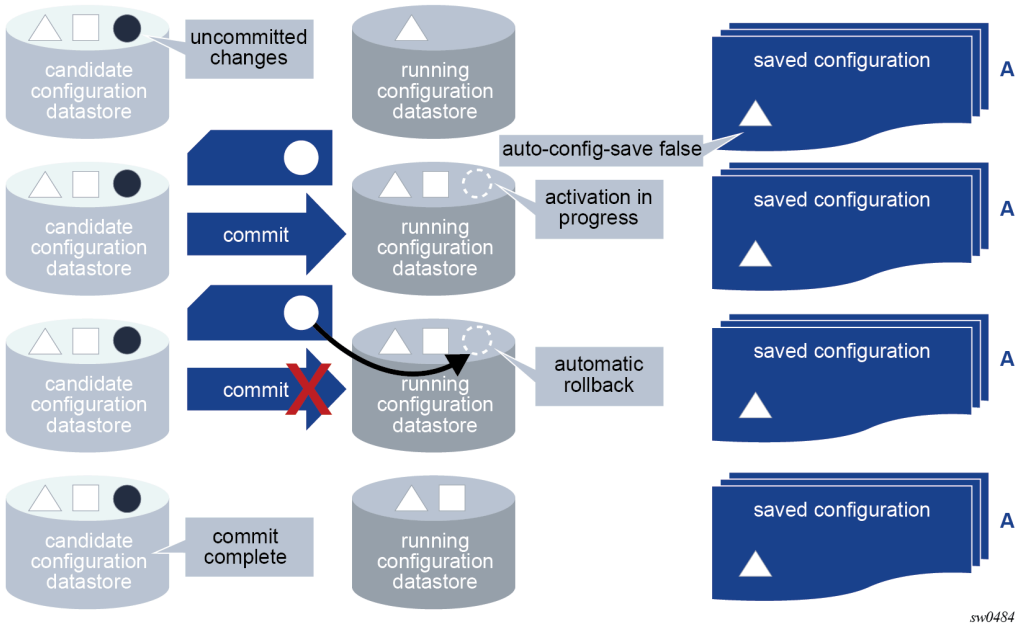
In the following figure, **auto-config-save** is set to **false**. The **admin save** command saves the running configuration before the commit is issued. However, a commit entry is created without saving the configuration.

Figure 8: Successful commit with auto-config-save false



In the following figure, the commit fails and no commit history entry or saved configuration file is created, regardless of the **auto-config-save** setting.

Figure 9: Failed commit



4.12 Loading configuration from a file or interactively

The **load** command loads the contents of an interactive scratchpad or a file into the candidate configuration. A configuration file can be local or remote.

The **load** command can only be executed at the top of the **configure** region when loading from a file. The contents of the file must start from the root of the configuration. The MD-CLI session must be in **private**, **exclusive**, or **global** configuration mode. Loading from a file does not result in a context change.

The **load** command is present working context (**pwc**) aware and can be executed from any context in the configuration tree. Interactive loading requires the input configuration to be correct from the pwc. See Loading configuration interactively for more information about interactive loading.

The **load** command can be executed regardless of whether uncommitted changes are present in the candidate configuration datastore.

The **full-replace** option replaces the current candidate configuration with the specified file. The **merge** option merges the contents of the specified file or interactive input into the candidate configuration. If there are conflicts, the configuration statements in the specified file or interactive input override the existing configuration statements. The configuration to be loaded is not the same as a CLI script to be executed and cannot include the following statements:

- MD-CLI commands such as **commit**, **delete**, or **tools**
- navigation commands such as **exit**, **back**, or **top**

See Executing commands from a file to perform such actions from a file.

If the loaded file or interactive input encounters errors, parsing terminates at the first error. When using the **load** command with a file, statements before the error are loaded into the candidate configuration. If an

error occurs using the **load** command interactively, no input configuration statements are incorporated into the candidate configuration.

Configuration statements in the loaded configuration are also subject to AAA command authorization. An authorization check failure terminates the execution of further statements in the input.

Note: If the router fails to boot because of an invalid configuration syntax, Nokia recommends to correct the syntax and reboot the router, which also reloads persistent indexes. This procedure is preferred over using **load full-replace** to restore the configuration without a reboot.

4.12.1 Using info outputs in load files

The plain text output from the **info full-context** or **info** commands can be copied and pasted into a load file. Both the **full-replace** and **merge** options support this type of content.

Example: Output from the info full-context command

This output can be copied and pasted into a file; for example, `cf3:\testbgp.cfg`.

```
*[ro:/configure router "Base" bgp]
A:admin@node-2# info full-context
    /configure router "Base" bgp neighbor "192.168.89.8" group "external"
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4 maximum 200
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4 log-only  true
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4    threshold 80

*[ro:/configure router "Base" bgp]
A:admin@node-2# info full-context > cf3:testbgp.cfg
```

```
[/]
A:admin@node-2# file list

Volume in drive cf3 on slot A is .

Volume in drive cf3 on slot A is formatted as FAT32

Directory of cf3:\

10/24/2019  04:02a      <DIR>          .ssh/
01/01/1980  12:00a                170 NVRAM.DAT
01/01/1980  12:00a                679 bof.cfg
10/24/2019  04:02a                314 nvsys.info
10/24/2019  04:02a                  1 restcntr.txt
10/27/2019  02:57p                257 testbgp.cfg
               5 File(s)                  1421 bytes.
               1 Dir(s)                      0 bytes free.
```

Example: From the MD-CLI, the file show command displays the contents of the file

```
[/]
A:admin@node-2# file show cf3:testbgp.cfg
File: testbgp.cfg
-------------------------------------------------------------------------------
    /configure router "Base" bgp neighbor "192.168.89.8" group "external"
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4 maximum  200
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4 log-only  true
    /configure router "Base" bgp neighbor "192.168.89.8" prefix-limit ipv4 threshold 80
```

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

Example: Using the load merge command

The **load merge** command can be used to merge the contents of the file into the candidate configuration. The following example shows no current candidate configuration changes for BGP before the command is executed. The **compare** command shows the candidate configuration changes after the file is merged.

```
[ex:/configure]
A:admin@node-2# load merge cf3:testbgp.cfg
Loaded 5 lines in 0.0 seconds from file cf3:\testbgp.cfg

*[ex:/configure]
A:admin@node-2# compare
    router "Base" {
        bgp {
+           group "external" {
+           }
+           neighbor "192.168.89.8" {
+               group "external"
+               prefix-limit ipv4 {
+                   maximum 200
+                   log-only true
+                   threshold 80
+               }
+           }
        }
    }
```

This output can also be copied into a file.

```
*[ro:/configure router "Base" bgp]
A:admin@node-2# info flat
    neighbor "192.168.89.8" group "external"
    neighbor "192.168.89.8" prefix-limit ipv4 maximum 200
    neighbor "192.168.89.8" prefix-limit ipv4 log-only true
    neighbor "192.168.89.8" prefix-limit ipv4 threshold 80

*[ro:/configure router "Base" bgp]
A:admin@node-2# info flat > cf3:testbgp.cfg

*[ro:/configure router "Base" bgp]
A:admin@node-2# file show cf3:testbgp.cfg
File: testbgp.cfg
-------------------------------------------------------------------------------
    neighbor "192.168.89.8" group "external"
    neighbor "192.168.89.8" prefix-limit ipv4 maximum 200
    neighbor "192.168.89.8" prefix-limit ipv4 log-only true
    neighbor "192.168.89.8" prefix-limit ipv4 threshold 80

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

Example: Adding a line to specify the context

An additional context line is added, through a manual edit, to specify the context **/configure router "Base" bgp**, as shown in the file display.

```
*[ro:/configure router "Base" bgp]
A:admin@node-2# file show cf3:testbgp.cfg
```

```
File: testbgp.cfg
--------------------------------------------------------------------------------
/configure router bgp
    neighbor "192.168.89.8" group "external"
    neighbor "192.168.89.8" prefix-limit ipv4 maximum 200
    neighbor "192.168.89.8" prefix-limit ipv4 log-only true
    neighbor "192.168.89.8" prefix-limit ipv4 threshold 80

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

Example: compare command shows candidate configuration changes

The file is merged and the **compare** command shows the resulting candidate configuration changes.

```
[ex:/configure router "Base" bgp]
A:admin@node-2# info

[ex:/configure router "Base" bgp]
A:admin@node-2# top

[ex:/configure]
A:admin@node-2# load merge cf3:testbgp.cfg
Loaded 6 lines in 0.0 seconds from file cf3:\testbgp.cfg

*[ex:/configure]
A:admin@node-2# compare
    router "Base" {
        bgp {
+           group "external" {
+           }
+           neighbor "192.168.89.8" {
+               group "external"
+               prefix-limit ipv4 {
+                   maximum 200
+                   log-only true
+                   threshold 80
+               }
+           }
        }
    }
```

Example: Output from the info command

To use the output in a load file, the context must be added through a manual edit, similar to the edit of the `testbgp.cfg` file in the preceding example, or use the output from the **info full-context** command.

```
*[ro:/configure router "Base" bgp]
A:admin@node-2# info
    group "external" {
    }
    neighbor "192.168.89.8" {
        group "external"
        prefix-limit ipv4 {
            maximum 200
            log-only true
            threshold 80
        }
    }
```

Example: Contents of the load file with the info output

```
/configure router "Base" bgp
    group "external" {
    }
    neighbor "192.168.89.8" {
        group "external"
        prefix-limit ipv4 {
            maximum 200
            log-only true
            threshold 80
        }
    }
```

4.12.2 Loading configuration interactively

The **load** command supports an interactive input scratchpad when operating in **merge** mode.

When the **load merge interactive** command is executed, a temporary scratchpad is presented to the user. Line numbers displayed on the left side of the input aid with troubleshooting if errors occur in the input.

The **interactive** option of the **merge** command can be used anywhere the **load** command is supported, including the root (/) of the **configure**, **debug**, or **li** region.

Example: Input prompt in an interactive load merge scratchpad

```
(pr)[/configure]
A:admin@node-2# load merge interactive
Interactive configuration input started. Syntax checking will be performed when
 configuration entry is complete.
Use Ctrl-D or enter EOF on a newline to conclude entry and start processing the entered
 configuration.
Use Ctrl-C to exit.
1>
```

All input is accepted into the scratchpad and processed at the end of the interactive session.

The input can be made in standard MD-CLI format, such as the output of the **info**, **info full-context**, or the **compare** command.

The interactive session ends in one of the following ways:

- Ctrl-C ends the interactive session and discards all input without processing
- Ctrl-D ends the interactive session and attempts to process the input and apply the input to the candidate configuration.
- **EOF** at the beginning of a new line without any other input on the line ends the interactive session and attempts to apply the input to the candidate configuration.

The system processes the input in the order of input. Parsing stops at the first error encountered and the system reports the error, referencing the line number in the input.

If no errors are found, the system merges the input configuration into the current candidate configuration. The changes take effect when the **commit** command is executed.

Example: load merge interactive session from /configure

```
[ex:/configure]
A:admin@node-2# load merge interactive
```

```
Interactive configuration input started. Syntax checking will be performed when
 configuration entry is complete.
Use Ctrl-D or enter EOF on a newline to conclude entry and start processing the entered
 configuration.
Use Ctrl-C to exit.
1> system
2>    name main_router
3>    contact "Head office"
4> EOF
Loaded 3 lines in 0.0 seconds

*[ex:/configure]
A:admin@node-2# compare
    system {
+       contact "Head office"
+       name "main_router"
    }
```

Example: load merge interactive session using compare output

```
[ex:/configure system]
A:admin@node-2# load merge interactive
Interactive configuration input started. Syntax checking will be performed when
 configuration entry is complete.
Use Ctrl-D or enter EOF on a newline to conclude entry and start processing the entered
 configuration.
Use Ctrl-C to exit.
1> +   contact "Head office"
2> -   name "node-2"
3> +   name "main_router"
4>
5> EOF
Loaded 4 lines in 0.0 seconds

*[ex:/configure system]
A:admin@node-2# compare
+   contact "Head office"
-   name "node-2"
+   name "main_router"
```

Example: load merge interactive session with errors

```
[ex:/configure]
A:admin@node-2# load merge interactive
Interactive configuration input started. Syntax checking will be performed when
 configuration entry is complete.
Use Ctrl-D or enter EOF on a newline to conclude entry and start processing the entered
 configuration.
Use Ctrl-C to exit.
1> system
2> name new_router
3> first incorrect entry "This flags an error"
4> second incorrect entry "Another error"
5> EOF
MINOR: MGMT_CORE #2201: Unknown element - 'first'
MINOR: MGMT_CORE #2502: Configuration load failed - error on line 3 - 'first incorrect
 entry "This flags an error"'
```

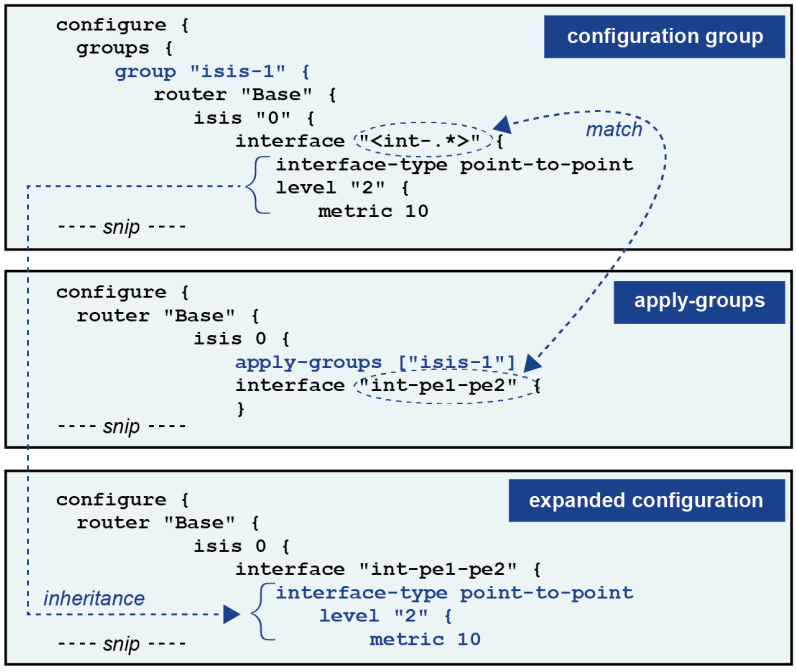
```
[ex:/configure router "Base"]
A:admin@node-2# load merge interactive
```

```
Interactive configuration input started. Syntax checking will be performed when
 configuration entry is complete.
Use Ctrl-D or enter EOF on a newline to conclude entry and start processing the entered
 configuration.
Use Ctrl-C to exit.
1> interface "base-1-1" {
2> port 1/1/3:1
3> ipv6 {
4> link-local-address {
5> address fe80::1
6> duplicate-address-detection false
7> }
8> address 2500::1 {
9> prefix-length 64
10> duplicate-address-detection false
11> }
12> vrrp 1 {
13> backup [2500::10 fe80::1:1]
14> message-interval 5
15> mac 00:00:5e:00:02:01
16> priority 130
17> ping-reply true
18> oper-group "op-v6LI-1"
19> bfd-liveness {
20> dest-ip 2000::2
21> service-name "100"
22> interface-name "bfd-1-1"
23> }
24> }
25> }
26> }
27>
28> Ctrl-D
MINOR: MGMT_CORE #2301: Invalid element value - 5 out of range 10..4095
MINOR: MGMT_CORE #2502: Configuration load failed - error on line 14 - 'message-interval
 5'
```

4.13 Using configuration groups

The MD-CLI supports the creation of configuration templates called configuration groups, which can be applied at different branches in the configuration, where the configuration elements are inherited. This is shown in the following figure.

Figure 10: Configuration groups



The advantage of using configuration groups is that similar configurations can be grouped in a template that is applied at multiple branches in the configuration tree. Subsequent configuration updates are only required in one location. Using groups, configurations can be organized in a logical fashion, such as regional (East vs West) or functional (core-facing vs access-facing parameters). The result is a more compact configuration that is easier to maintain and that reduces the number of configuration and operational errors.

Configuration groups are supported for all configuration branches and its descendants, which includes the configuration groups definition and applying the groups with the **apply-groups** command.

4.13.1 Creating configuration groups

Configuration groups are created in the groups branch of the configuration tree.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-backbone" {
            router "Base" {
                isis "0" {
                    interface "int-pe1-pe2" {
                        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
                        hello-authentication-type message-digest
                        hello-authentication true
                        interface-type point-to-point
                    }
                }
            }
        }
```

```
    }
```

Multiple configuration groups can be created, each with a unique name.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-access" {
            router "Base" {
                # configuration elements
            }
        }
        group "isis-backbone” {
            router "Base" {
                # configuration elements
            }
        }
        group "qos-backbone” {
            card "1" {
                # configuration elements
            }
            port "1/1/1" {
                # configuration elements
            }
            qos {
                # configuration elements
            }
            router "Base" {
                # configuration elements
            }
        }
    }
```

The configuration elements in a configuration group always start at a top-level configuration branch, such as **router**, **qos**, or **card**.

To match on a key of a list entry in a configuration group, an exact match or a regular expression match can be used.

4.13.1.1 Exact match

With an exact match, configuration elements can only be inherited by the list entry that matches the specified key value. When no list entry is matched, a new list entry is created with the specified key value.

In the following example, interface "int-pe1-pe2” is an exact match. When the group is applied and IS-IS interface "int-pe1-pe2” exists in IS-IS instance 0, the **interface-type** leaf is inherited. If the IS-IS interface does not exist, it is created with the **interface-type** set to **point-to-point**.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-backbone" {
            router "Base" {
                isis "0" {
                    interface "int-pe1-pe2" {
                        interface-type point-to-point
                    }
                }
            }
        }
```

```
    }
```

4.13.1.2 Regular expression match

With a regular expression match, configuration elements can be inherited by all list entries for which the key value matches the regular expression. A list entry cannot be created with a regular expression match.

In the following example, **interface "<.*>"** is a regular expression match that matches any interface name. When the group is applied, all configured IS-IS interfaces in IS-IS instance 0 inherit the **interface-type** leaf.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-backbone" {
            router "Base" {
                isis "0" {
                    interface "<.*>" {
                        interface-type point-to-point
                    }
                }
            }
        }
    }
```

4.13.1.2.1 Regular expression match format

A regular expression match is specified as a string with the regular expression enclosed in quotation marks and angle brackets: "<regex-match>".

The regular expression match is implicitly anchored: a ^ (match-starting position) is added at the beginning of the regular expression and a $ (match-ending position) is added at the end.

The regular expression is a subset of the Extended Regular Expression (ERE) notation as described in Using regular expressions.

For example:

- interface "<int-.*>" — matches all interfaces that start with "int-"
- interface "<.*>" — matches all interfaces
- interface "<.*pe[1-3].*>" — matches all interfaces that have "pe1", "pe2", or "pe3" in their name

Note:
A regular expression match on an encrypted leaf is restricted to a match all: "<.*>". Any other string enclosed in angle brackets ("<*string*>") is accepted as an exact match for the encrypted leaf and displayed as a hashed value in the configuration.

4.13.1.3 Conflicting match criteria within a configuration group

With a regular expression match, a match criteria conflict can occur if two regular expressions match or if a regular expression and an exact match both match on the same list entry. Conflicting matches within a configuration group are not supported and result in a validation error.

In the following configuration example, both **interface "<int-.*>"** and **interface "int-pe1-pe2"** match **isis 0 interface "int-pe1-pe2"**. At validation, this results in a configuration group inheritance failure because of conflicting match criteria:

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-backbone" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level-capability 2
                    }
                    interface "int-pe1-pe2" {
                        level 2 {
                            hello-interval 1
                        }
                    }
                }
            }
        }
    }

---snip---

    router "Base" {

---snip---

        isis 0 {
            apply-groups ["isis-backbone"]

---snip---
            interface "int-pe1-pe2" {
---snip---

           }
        }
    }

(ex)[configure]
A:admin@node-2# validate
MINOR: MGMT_CORE #2901: configure router "Base" isis 0 interface "int-pe1-pe2" - Configuration
 group inheritance failed - conflicting match criteria within group   "isis-backbone"
```

Conflicting match criteria within a configuration group can be avoided by applying multiple configuration groups.

```
*(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-backbone-common" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level-capability 2
                    }
                }
            }
        }
```

```
        group "isis-backbone-custom" {
            router "Base" {
                isis "0" {
                    interface "int-pe1-pe2" {
                        level 2 {
                            hello-interval 1
                        }
                    }
                }
            }
        }
    }

---snip---

    router "Base" {

---snip---

        isis 0 {
            apply-groups ["isis-backbone-custom" "isis-backbone-common"]

---snip---

            interface "int-pe1-pe2" {
---snip---
           }
        }
    }

*(ex)[configure router "Base" isis 0]
A:admin@node-2# validate

*(ex)[configure router "Base" isis 0]
A:admin@node-2# info inheritance
    apply-groups ["isis-backbone-custom" "isis-backbone-common"]
    interface "int-pe1-pe2" {
        ## 'interface-type' inherited from group "isis-backbone-common"
        interface-type point-to-point
        ## 'level-capability' inherited from group "isis-backbone-common"
        level-capability 2
        level 2 {
            ## 'hello-interval' inherited from group "isis-backbone-custom"
            hello-interval 1
        }
    }
```

4.13.2 Applying configuration groups

To inherit configuration elements from a configuration group, apply the group in a branch of the configuration tree with the **apply-groups** statement. For example:

```
(ex)[configure router "Base" isis 0]
A:admin@node-2# info
    apply-groups ["isis-1"]
```

Configuration elements from the corresponding branches where the group is applied are inherited. In the following example, the configuration group "isis-3" has configuration elements in both the **router isis**

interface and **router isis level** branch. Because the configuration group is applied at the **router isis interface** branch, only these configuration elements are inherited.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-3" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level "2" {
                            metric 30
                        }
                    }
                    level "2" {
                        wide-metrics-only true
                    }
                }
            }
        }
    }

---snip---

   router "Base" {
        isis 0 {
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                apply-groups ["isis-3"]
            }
        }
    }
```

The resulting expanded configuration can be shown with the **info inheritance** command:

```
(ex)[configure]
A:admin@node-2# info inheritance
   router "Base" {
        isis 0 {
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                apply-groups ["isis-3"]
                ## 'interface-type' inherited from group "isis-3"
interface-type point-to-point
                level 2 {
                    ## 'metric' inherited from group "isis-3"
metric 30
                }
            }
        }
    }
```

The following notes apply to configuration groups and the **apply-groups** statements:

- configuration groups cannot be nested; therefore, **apply-groups** statements cannot be part of a configuration group

- configuration groups that are not applied in the configuration do not functionally change the configuration
- configuration groups and **apply-groups** statements are part of the running configuration and are saved in the MD-CLI configuration file
- the **apply-groups** statement can be configured in the configuration root (that is, **/configure**) but is not displayed in the online help for the context

4.13.3 Inheritance rules

Local configuration elements have precedence over configuration group inheritance.

In the following example, the configuration group "isis-1" contains the configuration element **level-capability 1**, which is not inherited because a corresponding local configuration element exists.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-1" {
            router "Base" {
                isis "0" {
                    level-capability 1
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level "2" {
                            metric 10
                        }
                    }
                }
            }
        }
    }

---snip---

   router "Base" {
        isis 0 {
            apply-groups ["isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
            }
        }
    }
```

The resulting expanded configuration after inheritance is shown as follows:

```
(ex)[configure]
A:admin@node-2# info inheritance
   router "Base" {
        isis 0 {
            apply-groups ["isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                ## 'interface-type' inherited from group "isis-1"
interface-type point-to-point
```

```
                level 2 {
                    ## 'metric' inherited from group "isis-1"
metric 10
                }
            }
        }
    }
```

Up to eight configuration groups can be applied to a configuration branch. The configuration order determines the inheritance precedence:

- configuration elements in the first listed group have the highest precedence
- configuration elements in the last listed group have the lowest precedence

In the following example, both configuration groups "isis-1" and "isis-2" set an interface **level 2 metric**. Because configuration group "isis-2" is listed first in the **apply-groups**, its configuration elements have precedence. The **interface-type** configuration element is inherited from group "isis-1" because a corresponding configuration element is not present in group "isis-2" nor is it locally configured.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-1" {
            router "Base" {
                isis "0" {
                    level-capability 1
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level "2" {
                            metric 10
                        }
                    }
                }
            }
        }
        group "isis-2" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        level "2" {
                            metric 20
                        }
                    }
                }
            }
        }
    }

---snip---

   router "Base" {
        isis 0 {
            apply-groups ["isis-2" "isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
            }
        }
    }
```

The resulting expanded configuration after inheritance is shown as follows:

```
(ex)[configure]
A:admin@node-2# info inheritance
   router "Base" {
        isis 0 {
            apply-groups ["isis-2" "isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                ## 'interface-type' inherited from group "isis-1"
interface-type point-to-point
                level 2 {
                    ## 'metric' inherited from group "isis-2"
metric 20
                }
            }
        }
    }
```

Configuration groups can be applied at different hierarchical branches. The hierarchy determines the inheritance precedence.

Configuration elements in groups applied at a lower-level branch have precedence over configuration elements in groups applied at a higher-level branch.

In the following example, all configuration groups set an interface **level 2 metric**. Because configuration group "isis-3" is applied at the lowest level, its configuration elements have precedence. The **interface-type** configuration element is also inherited from group "isis-3" for the same reason. As described earlier, the **level-capability** configuration element from group "isis-1" has lower precedence than the local configured value. The **wide-metrics-only** configuration element from group "isis-3" is not inherited because the group is applied at the interface branch and only configuration elements at that level or lower can be inherited.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-1" {
            router "Base" {
                isis "0" {
                    level-capability 1
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level "2" {
                            metric 10
                        }
                    }
                }
            }
        }
        group "isis-2" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        level "2" {
                            metric 20
                        }
                    }
                }
            }
```

```
        }
        group "isis-3" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        interface-type point-to-point
                        level "2" {
                            metric 30
                        }
                    }
                    level "2" {
                        wide-metrics-only true
                    }
                }
            }
        }
    }

---snip---

   router "Base" {
        isis 0 {
            apply-groups ["isis-2" "isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                apply-groups ["isis-3"]
            }
        }
    }
```

The resulting expanded configuration after inheritance is shown as follows:

```
(ex)[configure]
A:admin@node-2# info inheritance
   router "Base" {
        isis 0 {
            apply-groups ["isis-2" "isis-1"]
            admin-state enable
            level-capability 2
            area-address [49.0001.0001]
            interface "int-pe1-pe2" {
                apply-groups ["isis-3"]
                ## 'interface-type' inherited from group "isis-3"
interface-type point-to-point
                level 2 {
                    ## 'metric' inherited from group "isis-3"
metric 30
                }
            }
        }
    }
```

Inheritance rules for leaf-lists are the same as for a single leaf, whether the list is a system-ordered leaf list (for example, **configure router interface** *interface-name* **if-attribute admin-group** *value*) or a user-ordered leaf list (for example, **configure router bgp export policy** *value*). The entire leaf-list is inherited, and it is not possible to add values to an existing leaf-list through configuration group inheritance.

Inheritance rules for user-ordered lists (for example, **configure policy-options policy-statement** *name* **named-entry** *entry-name*) are:

- list order is ignored for user-ordered list entry matching
- unmatched list entries in the configuration group definition and its descendant configuration elements are inherited in the locally-configured user-ordered list. Newly-created list entries are appended at the end in the order they appear in the configuration group definition.
- descendant configuration elements of matched user-ordered list entries are candidates for inheritance

4.13.4 Disabling inheritance

The **apply-groups-exclude** statement disables configuration inheritance for up to eight configuration groups in a configuration hierarchy. Configuration elements from the specified groups are not inherited in that branch. Disabling configuration inheritance for a configuration group only affects the configuration if that configuration group is applied at a higher-level branch in the configuration tree.

In the following example, configuration group "isis-1" is applied at the **/configure router "Base" isis 0** context, but inheritance for that group is disabled for interface "int-pe1-ce1".

```
[ex:configure groups]
A:admin@node-2# info
    group "isis-1" {
        router "Base" {
            isis 0 {
                interface "<int-.*>" {
                    interface-type point-to-point
                    level-capability 2
                }
                level 2 {
                    wide-metrics-only true
                }
            }
        }
    }

[ex:configure groups]
A:admin@node-2# /configure router isis

[ex:configure router "Base" isis 0]
A:admin@node-2# info
    apply-groups ["isis-1"]
    admin-state enable
    area-address [49.0001.0001]
    interface "int-pe1-ce1" {
        apply-groups-exclude ["isis-1"]
        passive true
    }
    interface "int-pe1-pe2" {
    }
    interface "int-pe1-pe3" {
    }
```

In the resulting expanded configuration, the interface parameters specified in the configuration group "isis-1" are not applied for interface "int-pe1-ce1":

```
[ex:configure router "Base" isis 0]
A:admin@node-2# info inheritance
    apply-groups ["isis-1"]
    admin-state enable
    area-address [49.0001.0001]
    interface "int-pe1-ce1" {
```

```
        apply-groups-exclude ["isis-1"]
        passive true
    }
    interface "int-pe1-pe2" {
        ## 'interface-type' inherited from group "isis-1"
        interface-type point-to-point
        ## 'level-capability' inherited from group "isis-1"
        level-capability 2
    }
    interface "int-pe1-pe3" {
        ## 'interface-type' inherited from group "isis-1"
        interface-type point-to-point
        ## 'level-capability' inherited from group "isis-1"
        level-capability 2
    }
    level 2 {
        ## 'wide-metrics-only' inherited from group "isis-1"
        wide-metrics-only true
    }
```

The following configuration guidelines apply to the **apply-groups-exclude** statement.

- This statement cannot be included in a configuration group.
- The statement is included in the running configuration and saved in the MD-CLI configuration file.
- The statement cannot be configured at the configuration root (**/configure**).

4.13.5 Displaying the expanded configuration

After configuring and applying configuration groups, the expanded configuration should be reviewed before the configuration is committed. The expanded configuration can be displayed with the **info inheritance** command. By default, this command displays the expanded candidate configuration. To display the expanded running configuration, use **info running inheritance**.

All statements that are inherited from a configuration group are tagged with a system comment.

```
(ex)[configure router "Base" isis 0 interface "int-pe1-pe2"]
A:admin@node-2# info inheritance
    ## 'interface-type' inherited from group "isis-1"
    interface-type point-to-point
    level 2 {
        ## 'metric' inherited from group "isis-2"
        metric 20
    }
```

Use the regular expression pattern match **info inheritance | match '^[ ]*##' invert-match** to suppress the system comments in the output of **info inheritance**.

```
(ex)[configure router "Base" isis 0 interface "int-pe1-pe2"]
A:admin@node-2# info inheritance | match '^[ ]*##' invert-match
    interface-type point-to-point
    level 2 {
        metric 20
    }
```

Note:
Conflicting matches are detected at validation. The **info inheritance** command may display an inherited configuration element in the candidate that is part of a conflicting match criteria.

The expanded running configuration can also be displayed with the **info intended** command. This command displays the intended configuration datastore as specified in RFC 8342, *Network Management Datastore Architecture* (NMDA) as follows:

- the **groups**, **apply-groups**, and **apply-groups-exclude** configuration statements are suppressed
- the expanded configuration with all statements that are inherited from a configuration group is displayed without the ## system comments

```
(ex)[configure router "Base" isis 0 interface "int-pe1-pe2"]
A:admin@node-2# info intended
    interface-type point-to-point
    level 2 {
        metric 20
    }
```

4.13.6 Authentication, Authorization, and Accounting (AAA) in configuration groups

User profiles can restrict the configuration branches that a user can change. Denied and read-only configuration branches in user profiles automatically prohibit inheritance of these branches via configuration groups.

To restrict a user from viewing, creating, or editing configuration branches inside a configuration group, an explicit entry for the configuration group must be added in the user profile.

In the following example, user admin2 has no access to the **sap-ingress** configuration branch.

```
(ex)[configure qos]
A:admin2@node-2# sap-ingress high-bw
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'sap-ingress'
```

This is enforced via the following entry in the local user profile:

```
(ro)[configure system security aaa local-profiles profile "restricted-admin"]
A:admin@node-2# info

---snip---

    entry 200 {
        match "configure qos sap-ingress"
        action deny
    }
```

The same entry prohibits configuration group inheritance for user admin2.

```
[ex:configure groups]
A:admin2@node-2# info
    group "grp-1" {
        qos {
            sap-ingress "high-bw" {
                queue 1 {
                    rate {
                        pir 200000
```

```
                    }
                }
                fc "be" {
                    queue 1
                }
            }
        }
    }
[ex:configure]
A:admin2@node-2# /configure qos apply-groups "grp-1"
MINOR: MGMT_CORE #2020: configure qos sap-ingress "high-bw" - Permission denied
```

User admin2 can still view, create, or change **sap-ingress** QoS policies inside a configuration group if the changes do not result in configuration group inheritance of the **sap-ingress** QoS policy.

User admin2 cannot see the result of configuration group inheritance of a **sap-ingress** QoS policy because the **info** command is subject to the AAA rules defined in the user profile.

```
[ex:configure groups]
A:admin2@node-2# info
    group "grp-1" {
        qos {
            sap-ingress "high-bw" {
                queue 1 {
                    rate {
                        pir 200000
                    }
                }
                fc "be" {
                    queue 1
                }
            }
        }
    }

[ex:configure groups]
A:admin2@node-2# /configure qos

[ex:configure qos]
A:admin2@node-2# info inheritance
    apply-groups ["grp-1"]
    md-auto-id {
        qos-policy-id-range {
            start 65000
            end 65535
        }
    }
```

An administrative user with full privileges can see the inherited configuration, which includes the **sap-ingress** QoS policy created by user admin2.

```
[ro:configure qos]
A:admin@node-2# info inheritance
    apply-groups ["grp-1"]
    md-auto-id {
        qos-policy-id-range {
            start 65000
            end 65535
        }
    }
    ## 'sap-ingress "high-bw"' inherited from group "grp-1"
    sap-ingress "high-bw" {
```

```
        ## 'queue 1' inherited from group "grp-1"
        queue 1 {
            ## 'rate' inherited from group "grp-1"
            rate {
                ## 'pir' inherited from group "grp-1"
                pir 200000
            }
        }
        ## 'fc "be"' inherited from group "grp-1"
        fc "be" {
            ## 'queue' inherited from group "grp-1"
            queue 1
        }
    }
```

To prevent user admin2 from viewing, creating, or changing **sap-ingress** QoS policies inside a configuration group, an explicit entry for the configuration group must be added to the AAA user profile.

```
[ex:configure system security aaa local-profiles profile "restricted-admin"]
A:admin@node-2# info

---snip---

    entry 200 {

        match "configure qos sap-ingress"
        action deny
    }
    entry 201 {
        match "configure groups group qos sap-ingress"
        action deny
    }
```

This configuration removes the privileges for user admin2 to view, create, or change **sap-ingress** QoS policies inside a configuration group.

```
[ex:configure groups]
A:admin2@node-2# info
    group "grp-1" {
        qos {
        }
    }

[ex:configure groups]
A:admin2@node-2# group "grp-1" qos sap-ingress "high-bw"
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'sap-ingress'
```

4.13.7 Configuration group example

The following configuration is an example of configuring IS-IS interface parameters using configuration groups.

In this example, all backbone IS-IS interface configuration parameters are part of the "isis-bb-interface" configuration group. A regular expression match "<int-.*>" is used to match on all backbone IS-IS interface names that start with "int-". The system loopback interface does not match the regular expression, so cannot inherit the configuration elements from the group.

The "isis-bb-interface" configuration group is applied at the router "Base", IS-IS instance 0 branch. When a new IS-IS backbone interface is added with a name that starts with "int-", it also inherits the configuration elements from the configuration group.

```
(ex)[configure]
A:admin@node-2# info
    groups {
        group "isis-bb-interface" {
            router "Base" {
                isis "0" {
                    interface "<int-.*>" {
                        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
                        hello-authentication-type message-digest
                        hello-padding adaptive
                        hello-authentication true
                        interface-type point-to-point
                    }
                }
            }
        }
    }

---snip---

    router "Base" {
        isis 0 {
            apply-groups ["isis-bb-interface"]
            admin-state enable
            ipv4-routing true
            ipv6-routing native
            level-capability 2
            area-address [49.0001.0001]
            multi-topology {
                ipv6-unicast true
            }
            interface "int-pe1-pe2" {
            }
            interface "int-pe1-pe3" {
            }
            interface "system" {
                passive true
            }
            level 2 {
                wide-metrics-only true
            }
        }
    }
```

The resulting expanded configuration after inheritance is shown as follows:

```
(ex)[configure router "Base" isis 0]
A:admin@node-2# info inheritance
    apply-groups ["isis-bb-interface"]
    admin-state enable
    ipv4-routing true
    ipv6-routing native
    level-capability 2
    area-address [49.0001.0001]
    multi-topology {
        ipv6-unicast true
    }
    interface "int-pe1-pe2" {
```

```
        ## 'hello-authentication-key' inherited from group "isis-bb-interface"
        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
        ## 'hello-authentication-type' inherited from group "isis-bb-interface"
        hello-authentication-type message-digest
        ## 'hello-padding' inherited from group "isis-bb-interface"
        hello-padding adaptive
        ## 'hello-authentication' inherited from group "isis-bb-interface"
        hello-authentication true
        ## 'interface-type' inherited from group "isis-bb-interface"
        interface-type point-to-point
    }
    interface "int-pe1-pe3" {
        ## 'hello-authentication-key' inherited from group "isis-bb-interface"
        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
        ## 'hello-authentication-type' inherited from group "isis-bb-interface"
        hello-authentication-type message-digest
        ## 'hello-padding' inherited from group "isis-bb-interface"
        hello-padding adaptive
        ## 'hello-authentication' inherited from group "isis-bb-interface"
        hello-authentication true
        ## 'interface-type' inherited from group "isis-bb-interface"
        interface-type point-to-point
    }
    interface "system" {
        passive true
    }
    level 2 {
        wide-metrics-only true
    }
```

The resulting expanded configuration after inheritance is shown as follows, without system comments:

```
(ex)[configure router "Base" isis 0]
A:admin@node-2# info inheritance | match '^[ ]*##' invert-match
    apply-groups ["isis-bb-interface"]
    admin-state enable
    ipv4-routing true
    ipv6-routing native
    level-capability 2
    area-address [49.0001.0001]
    multi-topology {
        ipv6-unicast true
    }
    interface "int-pe1-pe2" {
        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
        hello-authentication-type message-digest
        hello-padding adaptive
        hello-authentication true
        interface-type point-to-point
    }
    interface "int-pe1-pe3" {
        hello-authentication-key "KrbVPnF6Dg13PM/biw6ErHmrkAHk hash"
        hello-authentication-type message-digest
        hello-padding adaptive
        hello-authentication true
        interface-type point-to-point
    }
    interface "system" {
        passive true
    }
    level 2 {
        wide-metrics-only true
    }
```

4.13.8 Caveats

The following restrictions apply to configuration groups.

- Configuration groups are available only in model-driven management interface configuration mode and for configuration elements in the Nokia YANG models.
- When configuration groups are used with NETCONF, the <get-config> operation returns the pre-expanded configuration, including the configuration groups definitions and the **apply-groups** configuration elements. The expanded configuration, including the inherited configuration elements but excluding the configuration groups definitions and the **apply-groups** configuration elements, can be returned using the <get-data> operation on the intended datastore.
- When configuration groups are used with gNMI, the Get RPC command returns the pre-expanded configuration, including the configuration groups definitions and **apply-groups** configuration elements. The expanded configuration, including the inherited configuration elements, cannot be returned with gNMI.

For more information about NETCONF and gNMI, see the *7705 SAR Gen 2 System Management Guide*.

4.14 Viewing the status of the local datastores

An MD-CLI session in exclusive configuration mode acquires an explicit lock for both the global candidate and running configuration datastores. This is achieved by executing the **configure exclusive** command (in the implicit configuration workflow) or the **edit-config exclusive** command (in the explicit configuration workflow).

An explicit lock can also be obtained via:

- NETCONF or gRPC sessions. See the *7705 SAR Gen 2 System Management Guide* for more information.
- a private exclusive configuration session. See Private-exclusive configuration session for more information.

To view the lock status of the datastores, the following **show** command is available:

show system management-interface datastore-locks [**detail**]

Example: Using the detail option to display information

The **detail** option displays information about any model-driven interface session that impacts the datastore locks. MD-CLI read-only sessions, for example, do not impact the datastore locks.

```
(ro)[/]
A:admin@node-2# show system management-interface datastore-locks detail
===============================================================================
Session ID  Region                Datastore                    Lock State
   Username                       Session Mode                 Idle Time
   Session Type                   From
-------------------------------------------------------------------------------
 69         configure             Candidate, Running           Locked
   admin                          Exclusive                    0d 00:01:48
   MD-CLI                         192.168.144.87
-------------------------------------------------------------------------------
Number of sessions: 1
'#' indicates the current active session
```

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

Example: Using the configuration-sessions command to display information

The **configuration-sessions** command displays the same information as the **datastore-locks detail** command, but for all configuration sessions regardless of whether the session has a lock on the datastore.

```
(ro)[/]
A:admin@node-2# show system management-interface configuration-sessions
================================================================================
Session ID  Region                Datastore                     Lock State
   Username                       Session Mode                  Idle Time
   Session Type                   From
--------------------------------------------------------------------------------
#65         configure             Candidate                     Unlocked
   admin                          Private                       0d 00:00:00
   MD-CLI                         192.168.144.87
 66         configure             Candidate                     Unlocked
   admin                          Private                       0d 00:05:41
   MD-CLI                         192.168.144.87
 67         configure             Candidate                     Unlocked
   admin                          Private                       0d 00:05:08
   MD-CLI                         192.168.144.87
 68         configure             Candidate                     Unlocked
   admin                          Read-Only                     0d 00:02:25
   MD-CLI                         192.168.144.87
 69         configure             Candidate, Running            Locked
   admin                          Exclusive                     0d 00:01:54
   MD-CLI                         192.168.144.87
--------------------------------------------------------------------------------
Number of sessions: 5
'#' indicates the current active session
================================================================================
```

4.14.1 Unlocking a locked datastore

A datastore lock that has been acquired by any model-driven session can be administratively removed by using the following **admin** command:

admin disconnect session-id *session-id*

Example

For example, to disconnect the MD-CLI session indicated in the preceding **show** command output, issue the **admin** command as follows:

```
[/]
A:admin@node-2# admin disconnect session-id 10
```

Disconnecting an MD-CLI session (or any model-driven session, including NETCONF and gRPC) that acquired a datastore lock has the following results:

- Any uncommitted changes in the candidate configuration datastore are discarded.
- The session is terminated.
- The explicit lock is released.

4.15 Automatically running Python 3 applications before or after commit operations

SR OS allows Python 3 applications to be executed automatically: immediately before a commit, immediately after a commit, or both.

For more information, see the *7450 ESS, 7750 SR, 7950 XRS, and VSR System Management Guide*, "Executing Python automatically before or after commit operations" section.

5 Creating MD-CLI configuration from the classic CLI

Prerequisites

This section describes a procedure to convert classic CLI configuration snippets into MD-CLI configuration snippets on an SR OS node initially operating in classic configuration mode. The MD-CLI configuration snippets can be ported to another node operating in model-driven configuration mode or saved to a file for later use. For these snippet conversions, mixed configuration mode must be enabled.

When operating in mixed or model-driven interface configuration mode, the SR OS router automatically converts the running configuration to the MD-CLI format in memory. For a node operating in classic or mixed interface configuration mode, any additional configuration entered in the classic CLI can easily be converted to the MD-CLI format. Converting the classic CLI configuration avoids the need to manually recreate the configuration in the MD-CLI and ensures all applicable configuration elements are properly converted to the MD-CLI format by the system. Any Nokia SR OS router, including a virtual simulator, Virtualized Service Router (VSR), or lab router, can be used to convert configurations.

See the *7705 SAR Gen 2 System Management Guide*, section "Management Interface Configuration Mode" for more information about the management interface configuration mode.

The following example converts a configuration snippet which changes the system name.

5.1 Creating MD-CLI configuration from the classic CLI procedure

Procedure

Step 1. Ensure mixed management interface configuration mode is enabled and persistent (from the classic CLI engine) and log out of the current session.

Example

```
*A:node-2# /configure system management-interface configuration-mode mixed

Applying Changes to Model-Driven Database ... OK
*A:node-2# logout
```

Step 2. Log in to a new CLI session.

Example

```
Login: admin
Password:
```

Step 3. (Optional) Save the configuration in the classic CLI format. The saved configuration can be used to return to a known baseline configuration.

Example

```
*A:node-2# admin save
Writing configuration to cf3:config.cfg
Saving configuration ... OK
Completed.
```

Step 4. Switch to the MD-CLI engine and capture the configuration in the MD-CLI format to a file that can be used for later comparison. Because the **admin save** command is not supported in mixed configuration mode, use the **admin show configuration** command and redirect the output to the file.

Example

```
*A:node-2# //
INFO: CLI #2052: Switching to the MD-CLI engine

[]
A:admin@node-2# admin show configuration > cf3:md-config.cfg
```

Step 5. Return to the classic CLI engine and enter the configuration to be converted.

Example

```
[]
A:admin@node-2# //
INFO: CLI #2051: Switching to the classic CLI engine
A:node-2# configure system name new-node-7
```

Step 6. Switch to the MD-CLI engine. Compare the running configuration to the saved configuration file to display the additional configuration in the MD-CLI format.

Example

```
*A:new-node-7# //
INFO: CLI #2052: Switching to the MD-CLI engine

[]
A:admin@new-node-7# configure global
INFO: CLI #2054: Entering global configuration mode

[gl:configure]
A:admin@new-node-7# compare from url cf3:md-config.cfg
    system {
-       name "node-2"
+       name "new-node-7"
    }
```

Step 7. Copy and paste the differences in the MD-CLI engine on the target router or redirect to a file for later use. See Copying configuration elements and Using the file redirect option for more information.

5.1.1 MD-CLI configuration example output

The following example shows the conversion of a longer and more complex configuration that was pasted into the classic CLI using the preceding procedure.

Example

```
A:node-2>config>service# info
----------------------------------------------
        system
            bgp-auto-rd-range 2.2.2.2 comm-val 1 to 5000
        exit
        sdp 21 mpls create
```

```
            far-end 10.20.1.1
            ldp
            path-mtu 1600
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        customer 1 name "1" create
            description "Default customer"
        exit
        pw-template 100 name "100" create
            split-horizon-group "shg1"
            exit
        exit
        vpls 1 name "VPLS1" customer 1 vpn 1 create
            description "Vpls 1 "
            service-mtu 1400
            split-horizon-group "vpls1" create
                description "Default description for SHG vpls1"
            exit
            bgp
                route-distinguisher auto-rd
                route-target export target:100:1 import target:100:1
                pw-template-binding 100
                exit
            exit
            bgp-ad
                vpls-id 1:1
                no shutdown
            exit
            bgp-evpn
                evi 1
                mpls bgp 1
                    split-horizon-group "vpls1"
                    ingress-replication-bum-label
                    auto-bind-tunnel
                        resolution-filter
                            rsvp
                        exit
                        resolution filter
                    exit
                    no shutdown
                exit
            exit
            stp
                shutdown
            exit
            site "BGPMH1" create
                site-id 1
                split-horizon-group shg1
                failed-threshold 2
                no shutdown
            exit
            no shutdown
        exit
-----------------------------------------------
```

```
[gl:/configure]
A:admin@node-2# compare from url cf3:md-config.cfg
+   service {
+       pw-template "100" {
+           split-horizon-group {
```

```
+                name "shg1"
+            }
+        }
+        system {
+            bgp-auto-rd-range {
+                ip-address 2.2.2.2
+                community-value {
+                    start 1
+                    end 5000
+                }
+            }
+        sdp 21 {
+            admin-state enable
+            delivery-type mpls
+            path-mtu 1600
+            ldp true
+            far-end {
+                ip-address 10.20.1.1
+            }
+        }
+        vpls "VPLS1" {
+            admin-state enable
+            description "Vpls 1 "
+            service-id 1
+            customer "1"
+            vpn-id 1
+            service-mtu 1400
+            bgp 1 {
+                route-distinguisher auto-rd
+                route-target {
+                    export "target:100:1"
+                    import "target:100:1"
+                }
+                pw-template-binding "100" {
+                }
+            }
+            bgp-ad {
+                vpls-id "1:1"
+            }
+            bgp-evpn {
+                evi 1
+                mpls 1 {
+                    split-horizon-group "vpls1"
+                    ingress-replication-bum-label true
+                    auto-bind-tunnel {
+                        resolution filter
+                        resolution-filter {
+                            rsvp true
+                        }
+                    }
+                }
+            }
+            split-horizon-group "vpls1" {
+                description "Default description for SHG vpls1"
+            }
+            bgp-mh-site "BGPMH1" {
+                admin-state enable
+                id 1
+                failed-threshold 2
+                shg-name "shg1"
+            }
+        }
+    }
```

Note: Consider the following guidelines when converting classic CLI to MD-CLI configuration:

- Revert to a saved classic CLI configuration or save changes to the classic CLI configuration in each CLI engine when converting a configuration snippet so that the MD-CLI comparison displays only the snippet that is converted.
- Avoid configuration snippets that overlap with the existing configuration, including default configuration snippets, because they may not be displayed as a difference. A minimum configuration is recommended to ensure that all differences are displayed.
- Consider using the same type of router for the conversion as the router the configuration is intended for (although this is not a strict requirement), because some CLI commands are specific to a product family or chassis.

6 Switching between the classic CLI and the MD-CLI engines

A CLI command is available in both the classic CLI and MD-CLI engines to switch between the two engines in a user session. When authorized (**cli-engine** list contains both **classic-cli** and **md-cli**), the CLI engine switch command ("//", the double slash) can be executed from any CLI context in both engines to switch to the other CLI engine.

Example: Switching engines

```
A:node-2# //
INFO: CLI #2052: Switching to the MD-CLI engine

[/]
A:admin@node-2# //
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:node-2#
```

Example: Context saved when toggling

The context in which the CLI engine switch command is executed is saved when toggling between CLI engines and returns to the same context when toggling back.

```
[/]
A:admin@node-2# configure read-only
INFO: CLI #2066: Entering read-only configuration mode

[ro:/]
A:admin@node-2# configure router

[ro:/configure router "Base"]
A:admin@node-2# //
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:node-2# configure system management-interface
A:node-2>config>system>management-interface# //
INFO: CLI #2052: Switching to the MD-CLI engine

[ro:/configure router "Base"]
A:admin@node-2# //
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:node-2>config>system>management-interface#
```

Example: When switching is not authorized

If switching engines is not authorized (when **cli-engine** is only [**classic-cli**] or [**md-cli**]), the command is rejected.

```
A:node-2# //
MINOR: CLI #2053 Switching CLI engine is not authorized
```

```
A:node-2#
```

6.1 Executing classic CLI commands from the MD-CLI engine

When switching engines is authorized, all classic CLI engine commands can be executed from the MD-CLI engine. Entering a classic CLI engine command preceded by the "//" command executes the command in the classic CLI engine and returns immediately to the MD-CLI engine. The MD-CLI context is preserved before the switch to the classic CLI engine, and the context is restored when the session returns to the MD-CLI engine.

Example

In the following example, the classic CLI command is executed from the **configure system** context in the MD-CLI. When the session returns to the MD-CLI engine, it is returned to the same context.

```
[ex:/configure system]
A:admin@node-2# //show debug
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:admin-node-2# /show debug
debug
    router "Base"
        bgp
            open
        exit
    exit
exit
INFO: CLI #2052: Switching to the MD-CLI engine

[ex:/configure system]
A:admin@node-2#
```

It is acceptable to include a space between "//" and the CLI command. For example, the **//show debug** and **// show debug** commands are equivalent.

User interactions, such as pagination, confirmation, or control characters (for example, **Ctrl-C** to stop an ongoing command execution), are supported during CLI command execution. The CLI engine is switched back to the MD-CLI engine just before the CLI command prompt would normally appear.

Executing MD-CLI commands from the classic CLI engine works in the same way as described for executing classic CLI commands from the MD-CLI engine.

The following describes MD-CLI engine interactions with the classic CLI when using the "//" command:

- Uncommitted changes in the MD-CLI are kept when switching to the classic CLI.
- Command completion and ? help are not supported for the command following the "//".
- All control characters added on the same line when entering a "//" command have an effect on the CLI engine where they are entered.
- "//" or "//command" appears in the history of the CLI engine where it is executed.

6.2 Switching explicitly to the classic CLI engine

The **/!classic-cli** command is available in both the classic CLI and MD-CLI engines to explicitly switch to the classic CLI engine in a session, as long as **classic-cli** is an authorized CLI engine. If switching to the classic CLI engine is not authorized, the command is rejected. Issuing the**/!classic-cli** command in the classic CLI engine has no effect.

The **/!classic-cli** switch command can be executed from any CLI context in both engines and the context is preserved for both engines. When the command is executed, the session enters the last saved working context of the classic CLI engine.

Example: Executing the /!classic-cli switch command

```
A:node-2>config>system>management-interface# //
INFO: CLI #2052: Switching to the MD-CLI engine

[ex:/configure router "Base" bgp]
A:admin@node-2# /!classic-cli
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:node-2>#
```

6.3 Switching explicitly to the MD-CLI engine

The **/!md-cli** command is available in both the classic CLI and MD-CLI engines to explicitly switch to the MD-CLI engine in a session, as long as **md-cli** is an authorized CLI engine. If switching to the MD-CLI engine is not authorized, the command is rejected. Issuing the **/!md-cli** command in the MD-CLI engine has no effect.

The **/!md-cli** switch command can be executed from any CLI context in both engines and the context is preserved for both engines. When the command is executed, the session enters the last saved working context of the MD-CLI engine.

Example: Executing the /!md-cli switch command

```
[ex:/configure router "Base" bgp]
A:admin@node-2# /!classic-cli
INFO: CLI #2051: Switching to the classic CLI engine
INFO: CLI #2050: Classic CLI modification of the configuration is not allowed - 'model-
driven' management interface configuration mode active
A:node-2> /!md-cli
INFO: CLI #2052: Switching to the MD-CLI engine

[ex:/configure router "Base" bgp]
A:admin@node-2#
```

The **/!md-cli** and **/!classic-cli** commands can be useful when executing commands from a file, allowing the file to be executed in either CLI engine and ensuring the commands are run in the intended CLI engine.

7 Standards and protocol support

Note:
The information provided in this chapter is subject to change without notice and may not apply to all platforms.

Nokia assumes no responsibility for inaccuracies.

7.1 Bidirectional Forwarding Detection (BFD)

RFC 5880, *Bidirectional Forwarding Detection (BFD)*

RFC 5881, *Bidirectional Forwarding Detection (BFD) IPv4 and IPv6 (Single Hop)*

RFC 5882, *Generic Application of Bidirectional Forwarding Detection (BFD)*

7.2 Border Gateway Protocol (BGP)

draft-hares-idr-update-attrib-low-bits-fix-01, *Update Attribute Flag Low Bits Clarification*

draft-ietf-idr-add-paths-guidelines-08, *Best Practices for Advertisement of Multiple Paths in IBGP*

draft-ietf-idr-best-external-03, *Advertisement of the best external route in BGP*

draft-ietf-idr-bgp-gr-notification-01, *Notification Message support for BGP Graceful Restart*

draft-ietf-idr-bgp-optimal-route-reflection-10, *BGP Optimal Route Reflection (BGP-ORR)*

draft-ietf-idr-error-handling-03, *Revised Error Handling for BGP UPDATE Messages*

draft-ietf-idr-link-bandwidth-03, *BGP Link Bandwidth Extended Community*

RFC 1772, *Application of the Border Gateway Protocol in the Internet*

RFC 1997, *BGP Communities Attribute*

RFC 2385, *Protection of BGP Sessions via the TCP MD5 Signature Option*

RFC 2439, *BGP Route Flap Damping*

RFC 2545, *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*

RFC 2858, *Multiprotocol Extensions for BGP-4*

RFC 2918, *Route Refresh Capability for BGP-4*

RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*

RFC 4360, *BGP Extended Communities Attribute*

RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 4456, *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*

RFC 4486, *Subcodes for BGP Cease Notification Message*

RFC 4659, *BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*

RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*

RFC 4724, *Graceful Restart Mechanism for BGP* – helper mode

RFC 4760, *Multiprotocol Extensions for BGP-4*

RFC 4798, *Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)*

RFC 5004, *Avoid BGP Best Path Transitions from One External to Another*

RFC 5065, *Autonomous System Confederations for BGP*

RFC 5291, *Outbound Route Filtering Capability for BGP-4*

RFC 5396, *Textual Representation of Autonomous System (AS) Numbers* – asplain

RFC 5492, *Capabilities Advertisement with BGP-4*

RFC 5668, *4-Octet AS Specific BGP Extended Community*

RFC 6286, *Autonomous-System-Wide Unique BGP Identifier for BGP-4*

RFC 6368, *Internal BGP as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 6793, *BGP Support for Four-Octet Autonomous System (AS) Number Space*

RFC 6810, *The Resource Public Key Infrastructure (RPKI) to Router Protocol*

RFC 6811, *Prefix Origin Validation*

RFC 6996, *Autonomous System (AS) Reservation for Private Use*

RFC 7311, *The Accumulated IGP Metric Attribute for BGP*

RFC 7606, *Revised Error Handling for BGP UPDATE Messages*

RFC 7607, *Codification of AS 0 Processing*

RFC 7752, *North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP*

RFC 7911, *Advertisement of Multiple Paths in BGP*

RFC 7999, *BLACKHOLE Community*

RFC 8092, *BGP Large Communities Attribute*

RFC 8097, *BGP Prefix Origin Validation State Extended Community*

RFC 8212, *Default External BGP (EBGP) Route Propagation Behavior without Policies*

RFC 8277, *Using BGP to Bind MPLS Labels to Address Prefixes*

RFC 9294, *Application-Specific Link Attributes Advertisement Using the Border Gateway Protocol - Link State (BGP LS)*

RFC 9494, *Long-Lived Graceful Restart for BGP*

7.3 Bridging and management

IEEE 802.1AB, *Station and Media Access Control Connectivity Discovery*

IEEE 802.1ad, *Provider Bridges*

IEEE 802.1AX, *Link Aggregation*

IEEE 802.1D, *MAC Bridges*

IEEE 802.1p, *Traffic Class Expediting*

IEEE 802.1Q, *Virtual LANs*

IEEE 802.1s, *Multiple Spanning Trees*

IEEE 802.1w, *Rapid Reconfiguration of Spanning Tree*

7.4 Certificate management

RFC 4210, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*

RFC 4211, *Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 6712, *Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)*

RFC 7030, *Enrollment over Secure Transport*

RFC 7468, *Textual Encodings of PKIX, PKCS, and CMS Structures*

7.5 Ethernet VPN (EVPN)

draft-ietf-bess-evpn-ipvpn-interworking-14, *EVPN Interworking with IPVPN*

RFC 7432, *BGP MPLS-Based Ethernet VPN*

RFC 8214, *Virtual Private Wire Service Support in Ethernet VPN*

RFC 8365, *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*

RFC 8560, *Seamless Integration of Ethernet VPN (EVPN) with Virtual Private LAN Service (VPLS) and Their Provider Backbone Bridge (PBB) Equivalents*

RFC 9047, *Propagation of ARP/ND Flags in an Ethernet Virtual Private Network (EVPN)*

RFC 9135, *Integrated Routing and Bridging in Ethernet VPN (EVPN)*

RFC 9136, *IP Prefix Advertisement in Ethernet VPN (EVPN)*

RFC 9161, *Operational Aspects of Proxy ARP/ND in Ethernet Virtual Private Networks*

RFC 9251, *Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Proxies for Ethernet VPN (EVPN)*

7.6 gRPC Remote Procedure Calls (gRPC)

cert.proto version 0.1.0, *gNOI Certificate Management Service*

file.proto version 0.1.0, *gNOI File Service*

gnmi.proto version 0.8.0, *gNMI Service Specification*

gnmi_ext.proto, *gNMI Commit Confirmed Extension*

gnmi_ext.proto, *gNMI Config Subscription Extension*

gnmi_ext.proto, *gNMI Depth Extension*

system.proto version 1.0.0, *gNOI System Service*

tunnel.proto version 0.2, *gRPC Tunnel Service*

PROTOCOL-HTTP2, *gRPC over HTTP2*

7.7 Intermediate System to Intermediate System (IS-IS)

draft-ietf-isis-mi-02, *IS-IS Multi-Instance*

draft-ietf-lsr-igp-ureach-prefix-announce-01, *IGP Unreachable Prefix Announcement* – without U-Flag and UP-Flag

draft-kaplan-isis-ext-eth-02, *Extended Ethernet Frame Size Support*

ISO/IEC 10589:2002 Second Edition, *Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)*

RFC 1195, *Use of OSI IS-IS for Routing in TCP/IP and Dual Environments*

RFC 2973, *IS-IS Mesh Groups*

RFC 3359, *Reserved Type, Length and Value (TLV) Codepoints in Intermediate System to Intermediate System*

RFC 3719, *Recommendations for Interoperable Networks using Intermediate System to Intermediate System (IS-IS)*

RFC 3787, *Recommendations for Interoperable IP Networks using Intermediate System to Intermediate System (IS-IS)*

RFC 5120, *M-ISIS: Multi Topology (MT) Routing in IS-IS*

RFC 5130, *A Policy Control Mechanism in IS-IS Using Administrative Tags*

RFC 5301, *Dynamic Hostname Exchange Mechanism for IS-IS*

RFC 5302, *Domain-wide Prefix Distribution with Two-Level IS-IS*

RFC 5303, *Three-Way Handshake for IS-IS Point-to-Point Adjacencies*

RFC 5304, *IS-IS Cryptographic Authentication*

RFC 5305, *IS-IS Extensions for Traffic Engineering TE*

RFC 5306, *Restart Signaling for IS-IS* – helper mode

RFC 5308, *Routing IPv6 with IS-IS*

RFC 5309, *Point-to-Point Operation over LAN in Link State Routing Protocols*

RFC 5310, *IS-IS Generic Cryptographic Authentication*

RFC 6213, *IS-IS BFD-Enabled TLV*

RFC 6232, *Purge Originator Identification TLV for IS-IS*

RFC 6233, *IS-IS Registry Extension for Purges*

RFC 7775, *IS-IS Route Preference for Extended IP and IPv6 Reachability*

RFC 7794, *IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability* – sections 2.1 and 2.3

RFC 7981, *IS-IS Extensions for Advertising Router Information*

RFC 7987, *IS-IS Minimum Remaining Lifetime*

RFC 8202, *IS-IS Multi-Instance* – single topology

RFC 8570, *IS-IS Traffic Engineering (TE) Metric Extensions* – Min/Max Unidirectional Link Delay metric for flex-algo, RSVP, SR-TE

RFC 8919, *IS-IS Application-Specific Link Attributes*

7.8 Internet Protocol (IP) general

RFC 768, *User Datagram Protocol*

RFC 793, *Transmission Control Protocol*

RFC 854, *Telnet Protocol Specifications*

RFC 1350, *The TFTP Protocol (revision 2)*

RFC 2784, *Generic Routing Encapsulation (GRE)*

RFC 3164, *The BSD syslog Protocol*

RFC 4250, *The Secure Shell (SSH) Protocol Assigned Numbers*

RFC 4251, *The Secure Shell (SSH) Protocol Architecture*

RFC 4252, *The Secure Shell (SSH) Authentication Protocol* – publickey, password

RFC 4253, *The Secure Shell (SSH) Transport Layer Protocol*

RFC 4254, *The Secure Shell (SSH) Connection Protocol*

RFC 4511, *Lightweight Directory Access Protocol (LDAP): The Protocol*

RFC 4513, *Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms* – TLS

RFC 4632, *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*

RFC 5082, *The Generalized TTL Security Mechanism (GTSM)*

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2* – TLS client, RSA public key

RFC 5289, *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*

RFC 5425, *Transport Layer Security (TLS) Transport Mapping for Syslog* – RFC 3164 with TLS

RFC 5656, *Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer* – ECDSA

RFC 5925, *The TCP Authentication Option*

RFC 5926, *Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)*

RFC 6398, *IP Router Alert Considerations and Usage* – MLD

RFC 6528, *Defending against Sequence Number Attacks*

RFC 8446, *The Transport Layer Security (TLS) Protocol Version 1.3*

RFC 8907, *The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol*

7.9 Internet Protocol (IP) multicast

RFC 1112, *Host Extensions for IP Multicasting*

RFC 2236, *Internet Group Management Protocol, Version 2*

RFC 2365, *Administratively Scoped IP Multicast*

RFC 2375, *IPv6 Multicast Address Assignments*

RFC 2710, *Multicast Listener Discovery (MLD) for IPv6*

RFC 3376, *Internet Group Management Protocol, Version 3*

RFC 3446, *Anycast Rendevous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP)*

RFC 3590, *Source Address Selection for the Multicast Listener Discovery (MLD) Protocol*

RFC 3618, *Multicast Source Discovery Protocol (MSDP)*

RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*

RFC 4541, *Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches*

RFC 4604, *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*

RFC 4610, *Anycast-RP Using Protocol Independent Multicast (PIM)*

RFC 4611, *Multicast Source Discovery Protocol (MSDP) Deployment Scenarios*

RFC 5059, *Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)*

RFC 5186, *Internet Group Management Protocol Version 3 (IGMPv3) / Multicast Listener Discovery Version 2 (MLDv2) and Multicast Routing Protocol Interaction*

RFC 5384, *The Protocol Independent Multicast (PIM) Join Attribute Format*

RFC 7761, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*

RFC 8487, *Mtrace Version 2: Traceroute Facility for IP Multicast*

7.10 Internet Protocol (IP) version 4

RFC 791, *Internet Protocol*

RFC 792, *Internet Control Message Protocol*

RFC 826, *An Ethernet Address Resolution Protocol*

RFC 1034, *Domain Names - Concepts and Facilities*

RFC 1035, *Domain Names - Implementation and Specification*

RFC 1191, *Path MTU Discovery* – router specification

RFC 1519, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*

RFC 1812, *Requirements for IPv4 Routers*

RFC 1918, *Address Allocation for Private Internets*

RFC 2131, *Dynamic Host Configuration Protocol*; Relay only

RFC 2132, *DHCP Options and BOOTP Vendor Extensions* – DHCP

RFC 2401, *Security Architecture for Internet Protocol*

RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*

RFC 3046, *DHCP Relay Agent Information Option (Option 82)*

RFC 3768, *Virtual Router Redundancy Protocol (VRRP)*

RFC 4884, *Extended ICMP to Support Multi-Part Messages* – ICMPv4 and ICMPv6 Time Exceeded

7.11 Internet Protocol (IP) version 6

RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*

RFC 2529, *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels*

RFC 3122, *Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification*

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*

RFC 3587, *IPv6 Global Unicast Address Format*

RFC 3596, *DNS Extensions to Support IP version 6*

RFC 3633, *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*

RFC 3971, *SEcure Neighbor Discovery (SEND)*

RFC 4007, *IPv6 Scoped Address Architecture*

RFC 4191, *Default Router Preferences and More-Specific Routes* – Default Router Preference

RFC 4193, *Unique Local IPv6 Unicast Addresses*

RFC 4291, *Internet Protocol Version 6 (IPv6) Addressing Architecture*

RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*

RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*

RFC 5722, *Handling of Overlapping IPv6 Fragments*

RFC 5798, *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6* – IPv6

RFC 5952, *A Recommendation for IPv6 Address Text Representation*

RFC 6164, *Using 127-Bit IPv6 Prefixes on Inter-Router Links*

RFC 8021, *Generation of IPv6 Atomic Fragments Considered Harmful*

RFC 8200, *Internet Protocol, Version 6 (IPv6) Specification*

7.12 Internet Protocol Security (IPsec)

draft-ietf-ipsec-isakmp-mode-cfg-05, *The ISAKMP Configuration Method*

draft-ietf-ipsec-isakmp-xauth-06, *Extended Authentication within ISAKMP/Oakley (XAUTH)*

RFC 2401, *Security Architecture for the Internet Protocol*

RFC 2403, *The Use of HMAC-MD5-96 within ESP and AH*

RFC 2404, *The Use of HMAC-SHA-1-96 within ESP and AH*

RFC 2405, *The ESP DES-CBC Cipher Algorithm With Explicit IV*

RFC 2406, *IP Encapsulating Security Payload (ESP)*

RFC 2407, *IPsec Domain of Interpretation for ISAKMP (IPsec DoI)*

RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)*

RFC 2409, *The Internet Key Exchange (IKE)*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman group for Internet Key Exchange (IKE)*

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*

RFC 3947, *Negotiation of NAT-Traversal in the IKE*

RFC 3948, *UDP Encapsulation of IPsec ESP Packets*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec ESP*

RFC 4109, *Algorithms for Internet Key Exchange version 1 (IKEv1)*

RFC 4301, *Security Architecture for the Internet Protocol*

RFC 4303, *IP Encapsulating Security Payload*

RFC 4307, *Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)*

RFC 4308, *Cryptographic Suites for IPsec*

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

RFC 4543, *The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

RFC 4835, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*

RFC 4945, *The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2 and PKIX*

RFC 5019, *The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments*

RFC 5282, *Using Authenticated Encryption Algorithms with the Encrypted Payload of the IKEv2 Protocol*

RFC 5903, *ECP Groups for IKE and IKEv2*

RFC 5996, *Internet Key Exchange Protocol Version 2 (IKEv2)*

RFC 5998, *An Extension for EAP-Only Authentication in IKEv2*

RFC 6379, *Suite B Cryptographic Suites for IPsec*

RFC 6380, *Suite B Profile for Internet Protocol Security (IPsec)*

RFC 6960, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

RFC 7296, *Internet Key Exchange Protocol Version 2 (IKEv2)*

RFC 7321, *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*

RFC 7383, *Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation*

RFC 7427, *Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)*

RFC 8784, *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*

7.13 Label Distribution Protocol (LDP)

draft-pdutta-mpls-ldp-adj-capability-00, *LDP Adjacency Capabilities*

draft-pdutta-mpls-ldp-v2-00, *LDP Version 2*

RFC 3037, *LDP Applicability*

RFC 3478, *Graceful Restart Mechanism for Label Distribution Protocol* – helper mode

RFC 5036, *LDP Specification*

RFC 5283, *LDP Extension for Inter-Area Label Switched Paths (LSPs)*

RFC 5443, *LDP IGP Synchronization*

RFC 5561, *LDP Capabilities*

RFC 5919, *Signaling LDP Label Advertisement Completion*

7.14 Multiprotocol Label Switching (MPLS)

RFC 3031, *Multiprotocol Label Switching Architecture*

RFC 3032, *MPLS Label Stack Encoding*

RFC 3270, *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services* – E-LSP

RFC 3443, *Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks*

RFC 5332, *MPLS Multicast Encapsulations*

RFC 5884, *Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)*

RFC 6424, *Mechanism for Performing Label Switched Path Ping (LSP Ping) over MPLS Tunnels*

RFC 7308, *Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)*

RFC 7746, *Label Switched Path (LSP) Self-Ping*

7.15 Network Address Translation (NAT)

RFC 4787, *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*

RFC 5382, *NAT Behavioral Requirements for TCP*

RFC 5508, *NAT Behavioral Requirements for ICMP*

7.16 Network Configuration Protocol (NETCONF)

RFC 5277, *NETCONF Event Notifications*

RFC 6020, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*

RFC 6022, *YANG Module for NETCONF Monitoring*

RFC 6241, *Network Configuration Protocol (NETCONF)*

RFC 6242, *Using the NETCONF Protocol over Secure Shell (SSH)*

RFC 6243, *With-defaults Capability for NETCONF*

RFC 8071, *NETCONF Call Home and RESTCONF Call Home* – NETCONF

RFC 8342, *Network Management Datastore Architecture (NMDA)* – Startup, Candidate, Running and Intended datastores

RFC 8525, *YANG Library*

RFC 8526, *NETCONF Extensions to Support the Network Management Datastore Architecture* – <get-data> operation

7.17 Media Sanitization

NIST Special Publication 800-88 Revision 1, *Guidelines for Media Sanitization* – CF, MMC, SSD, SD, USB

7.18 Open Shortest Path First (OSPF)

RFC 1765, *OSPF Database Overflow*

RFC 2328, *OSPF Version 2*

RFC 3101, *The OSPF Not-So-Stubby Area (NSSA) Option*

RFC 3509, *Alternative Implementations of OSPF Area Border Routers*

RFC 3623, *Graceful OSPF Restart Graceful OSPF Restart* – helper mode

RFC 3630, *Traffic Engineering (TE) Extensions to OSPF Version 2*

RFC 4576, *Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 4577, *OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 5185, *OSPF Multi-Area Adjacency*

RFC 5243, *OSPF Database Exchange Summary List Optimization*

RFC 5250, *The OSPF Opaque LSA Option*

RFC 5309, *Point-to-Point Operation over LAN in Link State Routing Protocols*

RFC 5642, *Dynamic Hostname Exchange Mechanism for OSPF*

RFC 6549, *OSPFv2 Multi-Instance Extensions*

RFC 6987, *OSPF Stub Router Advertisement*

RFC 7471, *OSPF Traffic Engineering (TE) Metric Extensions* – Min/Max Unidirectional Link Delay metric for flex-algo, RSVP, SR-TE

RFC 7684, *OSPFv2 Prefix/Link Attribute Advertisement*

RFC 7770, *Extensions to OSPF for Advertising Optional Router Capabilities*

RFC 8920, *OSPF Application-Specific Link Attributes*

7.19 Path Computation Element Protocol (PCEP)

draft-alvarez-pce-path-profiles-04, *PCE Path Profiles*

draft-ietf-pce-binding-label-sid-15, *Carrying Binding Label/Segment Identifier (SID) in PCE-based Networks.* – MPLS binding SIDs

RFC 5440, *Path Computation Element (PCE) Communication Protocol (PCEP)*

RFC 8231, *Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*

RFC 8253, *PCEPS: Usage of TLS to Provide a Secure Transport for the Path Computation Element Communication Protocol (PCEP)*

RFC 8281, *PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model*

RFC 8408, *Conveying Path Setup Type in PCE Communication Protocol (PCEP) Messages*

RFC 8664, *Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing*

7.20 Pseudowire (PW)

draft-ietf-l2vpn-vpws-iw-oam-04, *OAM Procedures for VPWS Interworking*

MFA Forum 12.0.0, *Multiservice Interworking - Ethernet over MPLS*

MFA Forum 13.0.0, *Fault Management for Multiservice Interworking v1.0*

MFA Forum 16.0.0, *Multiservice Interworking - IP over MPLS*

RFC 3916, *Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)*

RFC 3985, *Pseudo Wire Emulation Edge-to-Edge (PWE3)*

RFC 4385, *Pseudo Wire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN*

RFC 4446, *IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)*

RFC 4447, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*

RFC 4448, *Encapsulation Methods for Transport of Ethernet over MPLS Networks*

RFC 5085, *Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires*

RFC 5659, *An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge*

RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*

RFC 6073, *Segmented Pseudowire*

RFC 6310, *Pseudowire (PW) Operations, Administration, and Maintenance (OAM) Message Mapping*

RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*

RFC 6575, *Address Resolution Protocol (ARP) Mediation for IP Interworking of Layer 2 VPNs*

RFC 6718, *Pseudowire Redundancy*

RFC 6829, *Label Switched Path (LSP) Ping for Pseudowire Forwarding Equivalence Classes (FECs) Advertised over IPv6*

RFC 6870, *Pseudowire Preferential Forwarding Status bit*

RFC 7023, *MPLS and Ethernet Operations, Administration, and Maintenance (OAM) Interworking*

RFC 7267, *Dynamic Placement of Multi-Segment Pseudowires*

RFC 7392, *Explicit Path Routing for Dynamic Multi-Segment Pseudowires* – ER-TLV and ER-HOP IPv4 Prefix

RFC 8395, *Extensions to BGP-Signaled Pseudowires to Support Flow-Aware Transport Labels*

7.21 Quality of Service (QoS)

RFC 2430, *A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)*

RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

RFC 2597, *Assured Forwarding PHB Group*

RFC 3140, *Per Hop Behavior Identification Codes*

RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*

7.22 Remote Authentication Dial In User Service (RADIUS)

draft-oscca-cfrg-sm3-02, *The SM3 Cryptographic Hash Function*

RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*

RFC 2866, *RADIUS Accounting*

RFC 3162, *RADIUS and IPv6*

RFC 6613, *RADIUS over TCP* – with TLS

RFC 6614, *Transport Layer Security (TLS) Encryption for RADIUS*

RFC 6929, *Remote Authentication Dial-In User Service (RADIUS) Protocol Extensions*

7.23 Resource Reservation Protocol - Traffic Engineering (RSVP-TE)

RFC 2702, *Requirements for Traffic Engineering over MPLS*

RFC 2747, *RSVP Cryptographic Authentication*

RFC 2961, *RSVP Refresh Overhead Reduction Extensions*

RFC 3097, *RSVP Cryptographic Authentication -- Updated Message Type Value*

RFC 3209, *RSVP-TE: Extensions to RSVP for LSP Tunnels*

RFC 4124, *Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering*

RFC 4125, *Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering*

RFC 4127, *Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering*

RFC 4561, *Definition of a Record Route Object (RRO) Node-Id Sub-Object*

RFC 5817, *Graceful Shutdown in MPLS and Generalized MPLS Traffic Engineering Networks*

7.24 Routing Information Protocol (RIP)

RFC 1058, *Routing Information Protocol*

RFC 2080, *RIPng for IPv6*

RFC 2082, *RIP-2 MD5 Authentication*

RFC 2453, *RIP Version 2*

7.25 Segment Routing (SR)

RFC 8287, *Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes*

RFC 8426, *Recommendations for RSVP-TE and Segment Routing (SR) Label Switched Path (LSP) Coexistence*

RFC 8476, *Signaling Maximum SID Depth (MSD) Using OSPF* – node MSD

RFC 8491, *Signaling Maximum SID Depth (MSD) Using IS-IS* – node MSD

RFC 8660, *Segment Routing with the MPLS Data Plane*

RFC 8661, *Segment Routing MPLS Interworking with LDP*

RFC 8665, *OSPF Extensions for Segment Routing*

RFC 8667, *IS-IS Extensions for Segment Routing*

RFC 8669, *Segment Routing Prefix Segment Identifier Extensions for BGP*

RFC 9256, *Segment Routing Policy Architecture*

RFC 9350, *IGP Flexible Algorithm*

7.26 Simple Network Management Protocol (SNMP)

draft-blumenthal-aes-usm-04, *The AES Cipher Algorithm in the SNMP's User-based Security Model* – CFB128-AES-192 and CFB128-AES-256

draft-ietf-isis-wg-mib-06, *Management Information Base for Intermediate System to Intermediate System (IS-IS)*

draft-ietf-mpls-ldp-mib-07, *Definitions of Managed Objects for the Multiprotocol Label Switching, Label Distribution Protocol (LDP)*

draft-ietf-mpls-lsr-mib-06, *Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base Using SMIv2*

draft-ietf-mpls-te-mib-04, *Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base*

draft-ietf-ospf-mib-update-08, *OSPF Version 2 Management Information Base*

draft-ietf-vrrp-unified-mib-06, *Definitions of Managed Objects for the VRRP over IPv4 and IPv6* – IPv6

ESO-CONSORTIUM-MIB revision 200406230000Z, *esoConsortiumMIB*

IANA-ADDRESS-FAMILY-NUMBERS-MIB revision 200203140000Z, *ianaAddressFamilyNumbers*

IANAifType-MIB revision 200505270000Z, *ianaifType*

IANA-RTPROTO-MIB revision 200009260000Z, *ianaRtProtoMIB*

IEEE8021-CFM-MIB revision 200706100000Z, *ieee8021CfmMib*

IEEE8021-PAE-MIB revision 200101160000Z, *ieee8021paeMIB*

IEEE8023-LAG-MIB revision 200006270000Z, *lagMIB*

LLDP-MIB revision 200505060000Z, *lldpMIB*

RFC 1157, *A Simple Network Management Protocol (SNMP)*

RFC 1212, *Concise MIB Definitions*

RFC 1215, *A Convention for Defining Traps for use with the SNMP*

RFC 1724, *RIP Version 2 MIB Extension*

RFC 1901, *Introduction to Community-based SNMPv2*

RFC 2021, *Remote Network Monitoring Management Information Base Version 2 using SMIv2*

RFC 2206, *RSVP Management Information Base using SMIv2*

RFC 2578, *Structure of Management Information Version 2 (SMIv2)*

RFC 2579, *Textual Conventions for SMIv2*

RFC 2580, *Conformance Statements for SMIv2*

RFC 2787, *Definitions of Managed Objects for the Virtual Router Redundancy Protocol*

RFC 2819, *Remote Network Monitoring Management Information Base*

RFC 2856, *Textual Conventions for Additional High Capacity Data Types*

RFC 2863, *The Interfaces Group MIB*

RFC 2864, *The Inverted Stack Table Extension to the Interfaces Group MIB*

RFC 2933, *Internet Group Management Protocol MIB*

RFC 3014, *Notification Log MIB*

RFC 3273, *Remote Network Monitoring Management Information Base for High Capacity Networks*

RFC 3410, *Introduction and Applicability Statements for Internet Standard Management Framework*

RFC 3430, *Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping*

RFC 3411, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*

RFC 3412, *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*

RFC 3413, *Simple Network Management Protocol (SNMP) Applications*

RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*

RFC 3415, *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*

RFC 3416, *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*

RFC 3417, *Transport Mappings for the Simple Network Management Protocol (SNMP)* – SNMP over UDP over IPv4

RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*

RFC 3419, *Textual Conventions for Transport Addresses*

RFC 3434, *Remote Monitoring MIB Extensions for High Capacity Alarms*

RFC 3584, *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*

RFC 3593, *Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals*

RFC 3635, *Definitions of Managed Objects for the Ethernet-like Interface Types*

RFC 3826, *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*

RFC 3877, *Alarm Management Information Base (MIB)*

RFC 4001, *Textual Conventions for Internet Network Addresses*

RFC 4022, *Management Information Base for the Transmission Control Protocol (TCP)*

RFC 4113, *Management Information Base for the User Datagram Protocol (UDP)*

RFC 4273, *Definitions of Managed Objects for BGP-4*

RFC 4292, *IP Forwarding Table MIB*

RFC 4293, *Management Information Base for the Internet Protocol (IP)*

RFC 4878, *Definitions and Managed Objects for Operations, Administration, and Maintenance (OAM) Functions on Ethernet-Like Interfaces*

RFC 7420, *Path Computation Element Communication Protocol (PCEP) Management Information Base (MIB) Module*

RFC 7630, *HMAC-SHA-2 Authentication Protocols in the User-based Security Model (USM) for SNMPv3*

7.27 Timing

RFC 3339, *Date and Time on the Internet: Timestamps*

RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*

RFC 8573, *Message Authentication Code for the Network Time Protocol*

7.28 Two-Way Active Measurement Protocol (TWAMP)

RFC 5357, *A Two-Way Active Measurement Protocol (TWAMP)* – server, unauthenticated mode

RFC 5938, *Individual Session Control Feature for the Two-Way Active Measurement Protocol (TWAMP)*

RFC 6038, *Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features*

RFC 8545, *Well-Known Port Assignments for the One-Way Active Measurement Protocol (OWAMP) and the Two-Way Active Measurement Protocol (TWAMP)* – TWAMP

RFC 8762, *Simple Two-Way Active Measurement Protocol* – unauthenticated

RFC 8972, *Simple Two-Way Active Measurement Protocol Optional Extensions* – unauthenticated

RFC 9503, *Simple Two-Way Active Measurement Protocol (STAMP) Extensions for Segment Routing Networks* – excluding Sections 3, 4.1.2 and 4.1.3

7.29 Virtual Private LAN Service (VPLS)

RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*

RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*

RFC 5501, *Requirements for Multicast Support in Virtual Private LAN Services*

RFC 6074, *Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*

RFC 7117, *Multicast in Virtual Private LAN Service (VPLS)*

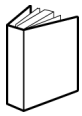
7.30 Yet Another Next Generation (YANG)

RFC 6991, *Common YANG Data Types*

RFC 7950, *The YANG 1.1 Data Modeling Language*

RFC 7951, *JSON Encoding of Data Modeled with YANG*

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback