CLI Usage

In This Chapter

This chapter provides information about using the command-line interface (CLI).

Topics in this chapter include:

CLI Structure

Alcatel-Lucent's SR OS CLI is a command-driven interface accessible through the console, Telnet and secure shell (SSH). The CLI can be used for configuration and management of SR OS routers.

The SR OS CLI command tree is a hierarchical inverted tree. At the highest level is the ROOT level. Below this level are other tree levels with the major command groups; for example, **configuration** commands and **show** commands are levels below ROOT.

The CLI is organized so related commands with the same scope are at the same level or in the same context. Sublevels or subcontexts have related commands with a more refined scope.

Figure 1 and Figure 2 examples display the major contexts for router configuration, and it is not a definitive list.

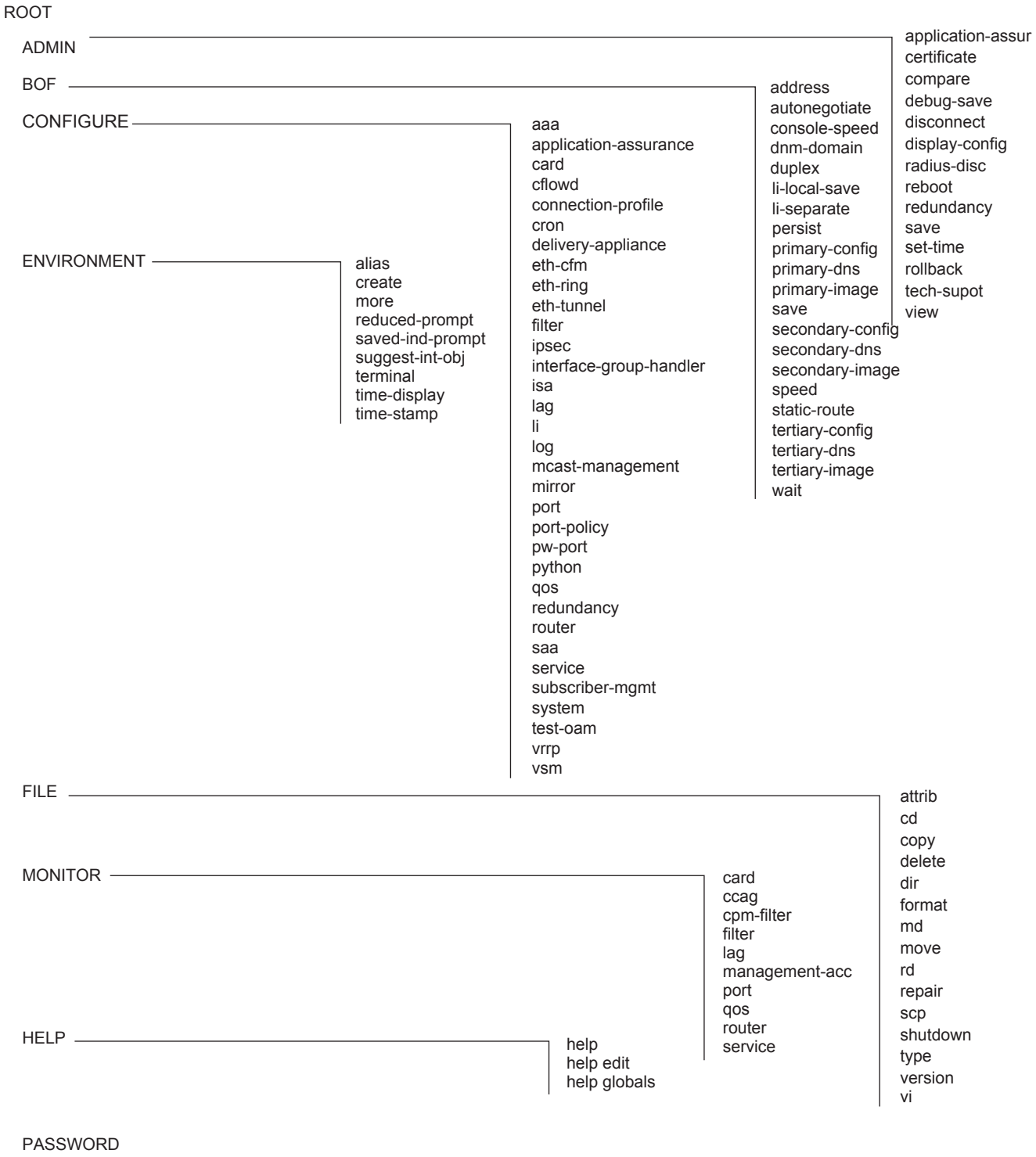

Figure 1: Root Commands

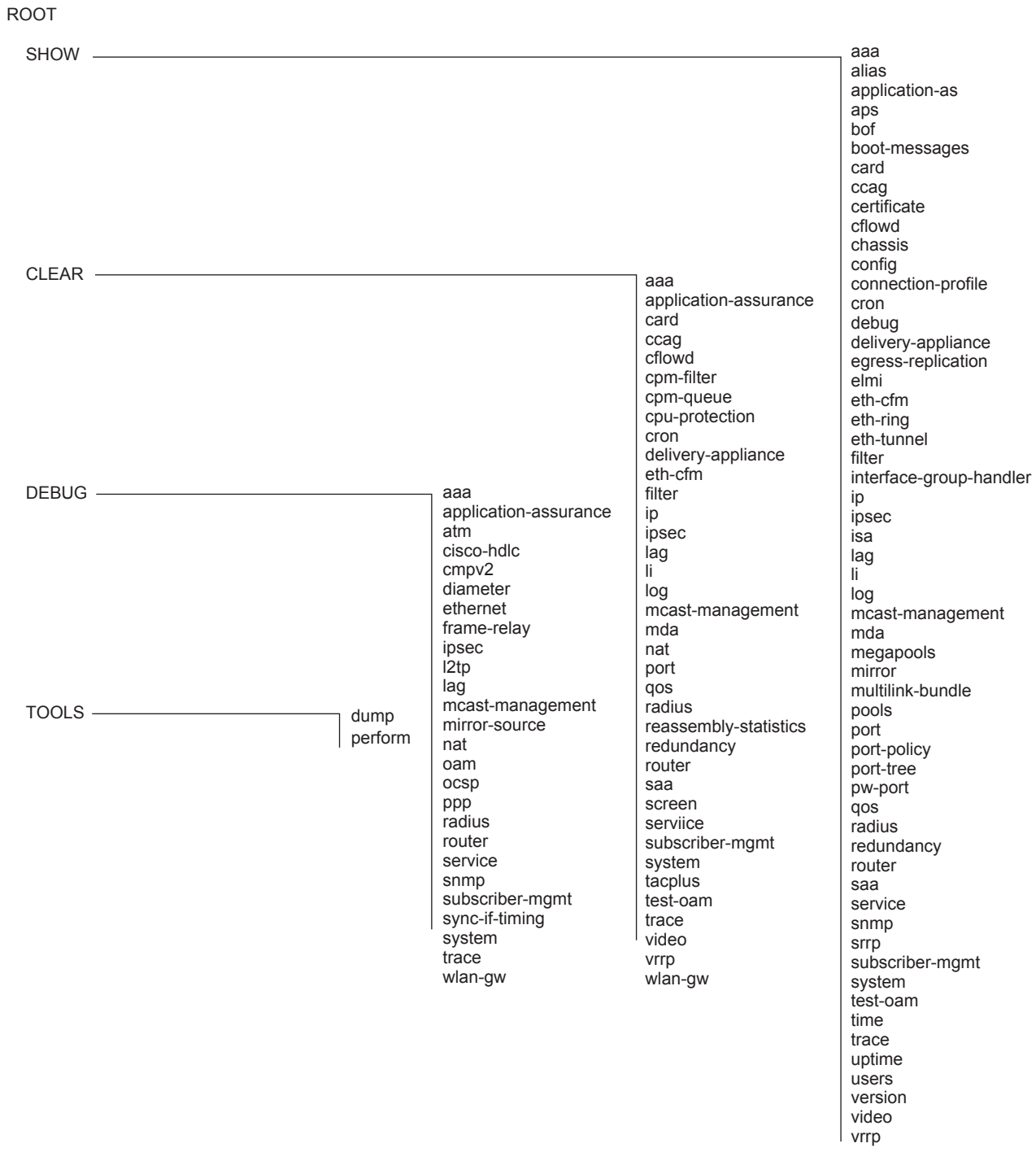

Figure 2: Operational Root Commands

Navigating in the CLI

The following sections describe additional navigational and syntax information.

CLI Contexts

Use the CLI to access, configure, and manage Alcatel-Lucent's SR OS routers. CLI commands are entered at the command line prompt. Access to specific CLI commands is controlled by the permissions set by your system administrator. Entering a CLI command makes navigation possible from one command context (or level) to another.

When you initially enter a CLI session, you are in the ROOT context. Navigate to another level by entering the name of successively lower contexts. For example, enter either the **configure** or **show** commands at the ROOT context to navigate to the **config** or **show** context, respectively. For example, at the command prompt (#), enter **config**. The active context displays in the command prompt.

```
A:ALA-12# config
A:ALA-12>config#
```

In a given CLI context, enter commands at that context level by simply entering the text. It is also possible to include a command in a lower context as long as the command is formatted in the proper command and parameter syntax.

The following example shows two methods to navigate to a service SDP ingress level:
Method 1:

```
A:ALA-12# configure service epipe 6 spoke-sdp 2:6 ingress
*A:ALA-12>config>service>epipe>spoke-sdp>ingress#
```

Method 2:

```
A:ALA-12>config# service
A:ALA-12>config>service# epipe 6
*A:ALA-12>config>service>epipe# spoke-sdp 2:6
*A:ALA-12>config>service>epipe>spoke-sdp# ingress
*A:ALA-12>config>service>epipe>spoke-sdp>ingress#
```

The CLI returns an error message when the syntax is incorrect.

```
*A:ALA-12>config# rooter
Error: Bad command.
```

Basic CLI Commands

The console control commands are the commands that are used for navigating within the CLI and displaying information about the console session. Most of these commands are implemented as global commands. They can be entered at any level in the CLI hierarchy with the exception of the `password` command which must be entered at the ROOT level. The console control commands are listed in Table 3.

Table 3: Console Control Commands

Command	Description	Page
`<Ctrl-c>`	Aborts the pending command.	
`<Ctrl-z>`	Terminates the pending command line and returns to the ROOT context.	
`back`	Navigates the user to the parent context.	100
`clear`	Clears statistics for a specified entity or clears and resets the entity.	100
`echo`	Echos the text that is typed in. Primary use is to display messages to the screen within an `exec` file.	101
`exec`	Executes the contents of a text file as if they were CLI commands entered at the console.	101
`exit`	Returns the user to the previous higher context.	101
`exit all`	Returns the user to the ROOT context.	102
`help` `?`	Displays help in the CLI.	103
`history`	Displays a list of the most recently entered commands.	104
`info`	Displays the running configuration for a configuration context.	105
`logout`	Terminates the CLI session.	107
`oam`	Provides OAM test suite options. See the OAM section of the SR OS OAM and Diagnostic Guide.	
`password`	Changes the user CLI login password. The password can only be changed at the ROOT level.	109
`ping`	Verifies the reachability of a remote host.	110
`pwc`	Displays the present or previous working context of the CLI session.	112

Table 3: Console Control Commands (Continued)

Command	Description	Page
sleep	Causes the console session to pause operation (sleep) for one second or for the specified number of seconds. Primary use is to introduce a pause within the execution of an exec file.	113
ssh	Opens a secure shell connection to a host.	113
telnet	Telnet to a host.	114
traceroute	Determines the route to a destination address.	114
tree	Displays a list of all commands at the current level and all sublevels.	115
write	Sends a console message to a specific user or to all users with active console sessions.	116

The list of all system global commands is displayed by entering `help globals` in the CLI. For example:

```
*A:ALA-12>config>service# help globals
back             - Go back a level in the command tree
      echo             - Echo the text that is typed in
      enable-admin     - Enable the user to become a system administrator
      exec             - Execute a file - use -echo to show the commands and
                         prompts on the screen
      exit             - Exit to intermediate mode - use option all to exit to
                         root prompt
      help             - Display help
      history          - Show command history
      info             - Display configuration for the present node
      logout           - Log off this system
      mrinfo           - Request multicast router information
      mstat            - Trace multicast path from a source to a receiver and
                         display multicast packet rate and loss information
      mtrace           - Trace multicast path from a source to a receiver
      oam              + OAM Test Suite
      ping             - Verify the reachability of a remote host
      pwc              - Show the present working context
      sleep            - Sleep for specified number of seconds
      ssh              - SSH to a host
      telnet           - Telnet to a host
      traceroute       - Determine the route to a destination address
      tree             - Display command tree structure from the context of
                         execution
      write            - Write text to another user
*A:ALA-12>config>service#
```

Table 4 lists describes command syntax symbols.

Table 4: Command Syntax Symbols

Symbol	Description
|	A vertical line indicates that one of the parameters within the brackets or braces is required. `tcp-ack {true|false}`
[ ]	Brackets indicate optional parameters. `redirects [number seconds]`
< >	Angle brackets indicate that you must enter text based on the parameter inside the brackets. `interface <interface-name>`
{ }	Braces indicate that one of the parameters must be selected. `default-action {drop|forward}`
[{ }]	Braces within square brackets indicates that you must choose one of the optional parameters. • **sdp** *sdp-id* [{**gre**|**mpls**}]
Bold	Commands in **bold** indicate commands and keywords.
Italic	Commands in *italics* indicate command options.

CLI Environment Commands

The CLI **environment** commands are found in the **root>environment** context of the CLI tree and controls session preferences for a single CLI session. The CLI **environment** commands are listed in Table 5.

Table 5: CLI Environment Commands

Command	Description	Page
`alias`	Enables the substitution of a command line by an alias.	117
`create`	Enables or disables the use of a create parameter check.	117
`more`	Configures whether CLI output should be displayed one screen at a time awaiting user input to continue.	117
`reduced-prompt`	Configures the maximum number of higher-level CLI context nodes to display by name in the CLI prompt for the current CLI session.	118
`saved-ind-prompt`	Saves the indicator in the prompt.	118
`suggest-internal-objects`	Enables the suggestion of internally created objects while auto completing.	119
`terminal`	Configures the terminal screen length for the current CLI session.	119
`time-display`	Specifies whether time should be displayed in local time or UTC.	119

CLI Monitor Commands

Monitor commands display specified statistical information related to the monitor subject (such as filter, port, QoS, router, service, and VRRP) at a configurable interval until a count is reached. The CLI **monitor** commands are found in the `root>monitor` context of the CLI tree.

The **monitor** command output displays a snapshot of the current statistics. The output display refreshes with subsequent statistical information at each configured interval and is displayed as a delta to the previous display.

The `<Ctrl-c>` keystroke interrupts a monitoring process. Monitor command configurations cannot be saved. You must enter the command for each monitoring session. Note that if the maximum limits are configured, you can monitor the statistical information for a maximum of 60 * 999 sec ~ 1000 minutes.

The CLI monitor command contexts are listed in Table 6.

Table 6: CLI Monitor Command Contexts

Command	Description	Page
card	Enables monitoring of ingress FP queue groups.	138
`ccag`	Enables CCAG port monitoring for traffic statistics.	122
`cpm-filter`	Monitor command output for CPM filters.	122
`filter`	Enables IP and MAC filter monitoring at a configurable interval until that count is reached.	124
`lag`	Enables Link Aggregation Group (LAG) monitoring to display statistics for individual port members and the LAG.	129
`management-access-filter`	Enables management access filter monitoring.	130
`port`	Enables port traffic monitoring. The specified port(s) statistical information displays at the configured interval until the configured count is reached.	132
`qos`	Enables arbiter and scheduler statistics monitoring.	136
`router`	Enables virtual router instance monitoring at a configurable interval until that count is reached.	147
`service`	Monitors commands for a particular service.	172

Getting Help in the CLI

The **help** system commands and the **?** key display different types of help in the CLI. Table 7 lists the different help commands.

Table 7: Online Help Commands

Command	Description
`help ?`	List all commands in the current context.
`string ?`	List all commands available in the current context that start with *string*.
`command ?`	Displays the command's syntax and associated keywords.
`command keyword ?`	List the associated arguments for *keyword* in *command*.
`string<Tab>`	Complete a partial command name (auto-completion) or list available commands that match *string*.

The **tree** and **tree detail** system commands are help commands useful when searching for a command in a lower-level context.

The following example displays a partial list of the **tree** and **tree detail** command output entered for the router node.

```
*A:cses-E11>config# tree
...
+---router
|   +---aggregate
|   +---allow-icmp-redirect
|   +---allow-icmp6-redirect
|   +---autonomous-system
|   +---bfd
|   |   +---abort
|   |   +---begin
|   |   +---bfd-template
|   |   |   +---echo-receive
|   |   |   +---multiplier
|   |   |   +---receive-interval
|   |   |   +---transmit-interval
|   |   |   +---type
|   |   +---commit
|   +---bgp
|   |   +---add-paths
|   |   |   +---ipv4
|   |   |   +---ipv6
|   |   |   +---vpn-ipv4
|   |   |   +---vpn-ipv6
|   |   +---advertise-external
|   |   +---advertise-inactive
|   |   +---aggregator-id-zero
|   |   +---auth-keychain
|   |   +---authentication-key
|   |   +---backup-path
|   |   +---best-path-selection
|   |   |   +---always-compare-med
|   |   |   +---as-path-ignore
|   |   |   +---deterministic-med
|   |   |   +---ignore-nh-metric
|   |   |   +---ignore-router-id
|   |   +---bfd-enable
|   |   +---cluster
...
```

```
*A:cses-E11>config# tree detail
...
+---router [<router-name>]
|   +---no aggregate <ip-prefix/ip-prefix-length>
|   |   aggregate <ip-prefix/ip-prefix-length> [summary-only] [as-set]
[aggregator <as-number:ip-address>] [black-hole [generate-icmp]]
[community <comm-id>]
|   |   aggregate <ip-prefix/ip-prefix-length> [summary-only] [as-set]
[aggregator <as-number:ip-address>] [community <comm-id>] [in-
direct <ip-address>]
|   +---allow-icmp-redirect
|   |   no allow-icmp-redirect
|   +---allow-icmp6-redirect
|   |   no allow-icmp6-redirect
|   +---autonomous-system <autonomous-system>
|   |   no autonomous-system
|   +---bfd
|   |   +---abort
|   |   +---begin
|   |   +---bfd-template <[32 chars max]>
|   |   |   no bfd-template <[32 chars max]>
|   |   |   +---echo-receive <milli-seconds>
|   |   |   |   no echo-receive
|   |   |   +---multiplier <[3..20]>
|   |   |   |   no multiplier
|   |   |   +---no receive-interval
|   |   |   |   receive-interval <milli-seconds>
|   |   |   +---no transmit-interval
|   |   |   |   transmit-interval <milli-seconds>
|   |   |   +---no type
|   |   |   |   type {cpm-np}
|   |   +---commit
|   +---bgp
|   |   no bgp
|   |   +---add-paths
|   |   |   no add-paths
|   |   |   +---ipv4 send <send-limit>
|   |   |   |   ipv4 send <send-limit> receive [none]
|   |   |   |   no ipv4
|   |   |   +---no ipv6
|   |   |   |   ipv6 send <send-limit>
|   |   |   |   ipv6 send <send-limit> receive [none]
|   |   |   +---no vpn-ipv4
|   |   |   |   vpn-ipv4 send <send-limit>
|   |   |   |   vpn-ipv4 send <send-limit> receive [none]
|   |   |   +---no vpn-ipv6
|   |   |   |   vpn-ipv6 send <send-limit>
|   |   |   |   vpn-ipv6 send <send-limit> receive [none]
|   |   +---advertise-external [ipv4] [ipv6]
|   |   |   no advertise-external [ipv4] [ipv6]
|   |   +---advertise-inactive
|   |   |   no advertise-inactive
|   |   +---aggregator-id-zero
|   |   |   no aggregator-id-zero
|   |   +---auth-keychain <name>
|       |       +---authentication-key   <authentication-key|hash-key>
[hash|hash2]
...
```

The CLI Command Prompt

By default, the CLI command prompt indicates the device being accessed and the current CLI context. For example, the prompt: **A:ALA-1>config>router>if#** indicates the active context, the user is on the device with hostname ALA-1 in the **configure>router>interface** context. In the prompt, the separator used between contexts is the ">" symbol.

At the end of the prompt, there is either a pound sign ("#") or a dollar sign ("$"). A "#" at the end of the prompt indicates the context is an existing context. A "$" at the end of the prompt indicates the context has been newly created. New contexts are newly created for logical entities when the user first navigates into the context.

Since there can be a large number of sublevels in the CLI, the **environment** command **reduced-prompt** *no of nodes in prompt* allows the user to control the number of levels displayed in the prompt.

All special characters (#, $, etc.) must be enclosed within double quotes, otherwise it is seen as a comment character and all characters on the command line following the # are ignored. For example:

```
*A:ALA-1>config>router# interface "primary#1"
```

When changes are made to the configuration file a "*" appears in the prompt string (*A:ALA-1) indicating that the changes have not been saved. When an admin save command is executed the "*" disappears. This behavior is controlled in the **saved-ind-prompt** command in the **environment** context.

Displaying Configuration Contexts

The `info` and `info detail` commands display configuration for the current level. The `info` command displays non-default configurations. The `info detail` command displays the entire configuration for the current level, including defaults. The following example shows the output that displays using the `info` command and the output that displays using the `info detail` command.

```
*A:ALA-1>config>router# interface system
*A:ALA-1>config>router>if# info
----------------------------------------------
            address 10.10.0.1/32
----------------------------------------------
*A:ALA-1>config>router>if#


*A:ALA-1>config>router>if# info detail
----------------------------------------------
            address 10.10.10.103/32 broadcast host-ones
            no description
            no arp-timeout
            no allow-directed-broadcasts
            tos-marking-state trusted
            no local-proxy-arp
            no proxy-arp
            icmp
                mask-reply
                redirects 100 10
                unreachables 100 10
                ttl-expired 100 10
            exit
            no mac
            no ntp-broadcast
            no cflowd
            no shutdown
----------------------------------------------
*A:ALA-1>config>router>if#
```

EXEC Files

The `exec` command allows you to execute a text file of CLI commands as if it were typed at a console device.

The `exec` command and the associated exec files can be used to conveniently execute a number of commands that are always executed together in the same order. For example, an `exec` command can be used by a user to define a set of commonly used standard command aliases.

The `echo` command can be used within an `exec` command file to display messages on screen while the file executes.

CLI Script Control

SR OS provides centralized script management for CLI scripts that are used by CRON and the Event Handling System (EHS). A set of script policies and script objects can be configured to control the following items and more:

- Where scripts are located (local compact flash, remote FTP server)
- Where to store the output of the results
- How long to keep historical script result records
- How long a script may run

If the scripts are located on local compact flash devices then the user must ensure that the scripts are on the compact flash devices of both CPMs so that operation of EHS continues as expected if a CPM switchover occurs.

A single script can be executing at one time. A table (SNMP smRunTable in the DISMAN-SCRIPT-MIB) is used as both an input queue of scripts waiting to be executed as well as for storage of records for completed scripts. If the input queue is full then the script request is discarded.

Entering CLI Commands

Command Completion

The CLI supports both command abbreviation and command completion. If the keystrokes entered are enough to match a valid command, the CLI displays the remainder of the command syntax when the <Tab> key or space bar is pressed. When typing a command, the <Tab> key or space bar invokes auto-completion. If the keystrokes entered are definite, auto-completion will complete the command. If the letters are not sufficient to identify a specific command, pressing the <Tab> key or space bar will display commands matching the letters entered.
System commands are available in all CLI context levels.

Unordered Parameters

In a given context, the CLI accepts command parameters in any order as long as the command is formatted in the proper command keyword and parameter syntax. Command completion will still work as long as enough recognizable characters of the command are entered.

The following output shows different **static-route** command syntax and an example of the command usage.

```
*A:ALA-12>config>router# static-route ?
- [no] static-route {<ip-prefix/mask>|<ip-prefix> <netmask>} [preference <preference>]
    [metric <metric>] [tag <tag>] [enable|disable] next-hop <ip-address|ip-int-name>
  - [no] static-route {<ip-prefix/mask>|<ip-prefix> <netmask>} [preference <preference>]
    [metric <metric>] [tag <tag>] [enable|disable] indirect <ip-address> [ldp
    [disallow-igp]]
  - [no] static-route {<ip-prefix/mask>|<ip-prefix> <netmask>} [preference <preference>]
    [metric <metric>] [tag <tag>] [enable|disable] black-hole
*A:ALA-12>config>router# static-route preference 1 10.1.0.0/16 metric
```

Editing Keystrokes

When entering a command, special keystrokes allow for editing of the command. Table 8 lists the command editing keystrokes.

Table 8: Command Editing Keystrokes

Editing Action	Keystrokes
Delete current character	<Ctrl-d>
Delete text up to cursor	<Ctrl-u>
Delete text after cursor	<Ctrl-k>
Move to beginning of line	<Ctrl-a>
Move to end of line	<Ctrl-e>
Get prior command from history	<Ctrl-p>
Get next command from history	<Ctrl-n>
Move cursor left	<Ctrl-b>
Move cursor right	<Ctrl-f>
Move back one word	<Esc><b>
Move forward one word	<Esc><f>
Convert rest of word to uppercase	<Esc><c>
Convert rest of word to lowercase	<Esc><l>
Delete remainder of word	<Esc><d>
Delete word up to cursor	<Ctrl-w>
Transpose current and previous character	<Ctrl-t>
Enter command and return to root prompt	<Ctrl-z>
Refresh input line	<Ctrl-l>

Absolute Paths

CLI commands can be executed in any context by specifying the full path from the CLI root. To execute an out-of-context command enter a forward slash "/" or backward slash "\" at the beginning of the command line. The forward slash "/" or backward slash "\" cannot be used with the **environment alias** command.The commands are interpreted as absolute path. Spaces between the slash and the first command will return an error. Commands that are already global (such as ping, telnet, exit, back, etc.) cannot be executed with a forward slash "/" or backward slash "\" at the beginning of the command line.

```
*A:ALA-12# configure router
*A:ALA-12>config>router# interface system address 1.2.3.4
*A:ALA-12>config>router# /admin save
*A:ALA-12>config>router# \clear router interface
*A:ALA-12>config>router#
```

The command may or may not change the current context depending on whether or not it is a leaf command. This is the same behavior the CLI performs when CLI commands are entered individually, for example:

```
*A:ALA-12# admin
*A:ALA-12>admin# save
```

or

```
*A:ALA-12# admin save
*A:ALA-12#
```

Note that an absolute path command behaves the same as manually entering a series of command line instructions and parameters.

For example, beginning in an IES context service ID 4 (IES 4),

CLI Syntax: `config>service>ies> /clear card 1`

behaves the same as the following series of commands.

Example:
```
config>service>ies>exit all
clear card 1
configure service ies 4 (returns you to your starting point)
config>service>ies
```

If the command takes you to a different context, the following occurs:

CLI Syntax: `config>service>ies>/configure service ies 5 create`

becomes

Example:
```
config>service>ies>exit all
configure service vpls 5 create
config>service>vpls>
```

History

The CLI maintains a history of the most recently entered commands. The `history` command displays the most recently entered CLI commands.

```
*A:ALA-1# history
   1 environment terminal length 48
   2 environment no create
   3 show version
   4 configure port 1/1/1
   5 info
   6 \configure router isis
   7 \port 1/1/2
   8 con port 1/1/2
   9 \con port 1/1/2
  10 \configure router bgp
  11 info
  12 \configure system login-control
  13 info
  14 history
  15 show version
  16 history
*A:ALA-1# !3
```

Entering Numerical Ranges

The SR OS CLI allows the use of a single numerical range as an argument in the command line. A range in a CLI command is limited to positive integers and is denoted with two numbers enclosed in square brackets with two periods ("..") between the numbers:

$$[x..y]$$

where *x* and *y* are positive integers and *y*-*x* is less than 1000.

For example, it is possible to shut down ports 1 through 10 in Slot 1 on MDA 1. A port is denoted with "*slot*/*mda*/*port*", where *slot* is the slot number, *mda* is the MDA number and *port* is the port number. To shut down ports 1 through 10 on Slot 1 and MDA 1, the command is entered as follows:

```
configure port 1/1/[1..10] shutdown
```

<Ctrl-C> can be used to abort the execution of a range command.

Specifying a range in the CLI does have limitations. These limitations are summarized in Table 9.

Table 9: CLI Range Use Limitations

Limitation	Description
Only a single range can be specified.	It is not possible to shut down ports 1 through 10 on MDA 1 and MDA 2, as the command would look like `configure port 1/[1..2]/[1..10]` and requires two ranges in the command, [1..2] for the MDA and [1..10] for the port number.

Ranges within quotation marks are interpreted literally.	In the CLI, enclosing a string in quotation marks (`"string"`) causes the string to be treated literally and as a single parameter. For example, several commands in the CLI allow the configuration of a descriptive string. If the string is more than one word and includes spaces, it must be enclosed in quotation marks. A range that is enclosed in quotes is also treated literally. For example,
	`configure router interface "A[1..10]" no shutdown`
	creates a single router interface with the name “A[1..10]”. However, a command such as:
	`configure router interface A[1..10] no shutdown`
	creates 10 interfaces with names A1, A2 .. A10.

The range cannot cause a change in contexts.	Commands should be formed in such a way that there is no context change upon command completion. For example, `configure port 1/1/[1..10]` will attempt to change ten different contexts. When a range is specified in the CLI, the commands are executed in a loop. On the first loop execution, the command changes contexts, but the new context is no longer valid for the second iteration of the range loop. A “Bad Command” error is reported and the command aborts.
Command completion may cease to work when entering a range.	After entering a range in a CLI command, command and key completion, which normally occurs by pressing the <Tab> or spacebar, may cease to work. If the command line entered is correct and unambiguous, the command works properly; otherwise, an error is returned.

Pipe/Match

The SR OS supports the pipe feature to search one or more files for a given character string or pattern.

Note: When using the pipe/match command the variables and attributes must be spelled correctly. The attributes following the command and must come before the expression/pattern. The following displays examples of the pipe/match command to complete different tasks:

- Task: Capture all the lines that include “echo” and redirect the output to a file on the compact flash:

  admin display-config | match “echo” > cf1:\test\echo_list.txt
- Task: Display all the lines that do not include “echo”:

  admin display-config | match invert-match “echo”
- Task: Display the first match of “vpls” in the configuration file:

  admin display-config | match max-count 1 “vpls”
- Task: Display everything in the configuration after finding the first instance of “interface”:

  admin display-config | match post-lines 999999 interface
- Task: Display a count of the total number of lines of output instead of displaying the output itself.

  admin display-config | match interface | count

Command syntax:

match *pattern* **context** {**parents** | **children** | **all**} [**ignore-case**] [**max-count** *lines-count*] [**expression**]

match *pattern* [**ignore-case**] [**invert-match**] [**pre-lines** *pre-lines*] [**post-lines** *lines-count*] [**max-count** *lines-count*] [**expression**]

where:

```
pattern         string or regular expression
context         keyword:  display context associated with the matching line
parents         keyword:  display parent context information
children        keyword:  display child context information
all             keyword:  display both parent and child context information
ignore-case     keyword
max-count       keyword:  display only a specific number of instances of matching lines
lines-count     1 — 2147483647
expression      keyword:  pattern is interpreted as a regular expression
invert-match    keyword
pre-lines       keyword:  display some lines prior to the matching line
pre-lines       0 — 100
post-lines      keyword:  display some lines after the matching line
lines-count     1 — 2147483647
```

For example:

```
A:Dut-C# show log log-id 98 | match ignore-case "sdp bind"
"Status of SDP Bind 101:1002 in service 1001 (customer 1) changed to admin=up oper=up
flags="
"Processing of a SDP state change event is finished and the status of all affected SDP
Bindings on SDP 101 has been updated."


A:Dut-C# show log log-id 98 | match max-count 1 "service 1001"
"Status of service 1001 (customer 1) changed to administrative state: up, operational
state: up"



A:Dut-C# admin display-config | match post-lines 5 max-count 2 expression "OSPF.*Config"
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
            timers
                spf-wait 1000 1000 1000
            exit
echo "OSPFv2 (Inst: 1) Configuration"
#--------------------------------------------------
        ospf 1
            asbr
            router-id 1.0.0.1
            export "testall"
*A:Dut# admin display-config | match debug_mirror
            profile "debug_mirror"


*A:Dut# admin display-config | match context parent debug_mirror
#--------------------------------------------------
    system
        security
            profile "debug_mirror"


*A:Dut# admin display-config | match context all debug_mirror
#--------------------------------------------------
    system
        security
            profile "debug_mirror"
                default-action deny-all
                entry 10
                exit


*A:Dut# show log event-control | match ignore-case pre-lines 10 SyncStatus
L  2016 tmnxLogOnlyEventThrottled        MA  gen          0           0
MCPATH:
   2001 tmnxMcPathSrcGrpBlkHole          MI  gen          0           0
   2002 tmnxMcPathSrcGrpBlkHoleClear     MI  gen          0           0
   2003 tmnxMcPathAvailBwLimitReached    MI  gen          0           0
   2004 tmnxMcPathAvailBwValWithinRange  MI  gen          0           0
MC_REDUNDANCY:
   2001 tmnxMcRedundancyPeerStateChanged WA  gen          0           0
   2002 tmnxMcRedundancyMismatchDetected WA  gen          0           0
   2003 tmnxMcRedundancyMismatchResolved WA  gen          0           0
   2004 tmnxMcPeerSyncStatusChanged      WA  gen          0           0
```

Table 10 describes regular expression symbols and interpretation (similar to what is used for route policy regexp matching). Table 11 describes special characters.

Table 10: Regular Expression Symbols

String	Description
.	Matches any single character.
[ ]	Matches a single character that is contained within the brackets. [abc] matches “a”, “b”, or “c”. [a-z] matches any lowercase letter. [A-Z] matches any uppercase letter. [0-9] matches any number.
[^]	Matches a single character that is not contained within the brackets. [^abc] matches any character other than “a”, “b”, or “c”. [^a-z] matches any single character that is not a lowercase letter.
^	Matches the start of the line (or any line, when applied in multiline mode)
$	Matches the end of the line (or any line, when applied in multiline mode)
()	Define a “marked subexpression”. Every matched instance will be available to the next command as a variable.
*	A single character expression followed by “*” matches zero or more copies of the expression.
{m,n}	Matches least m and at most n repetitions of the term
{m}	Matches exactly m repetitions of the term
{m,}	Matches m or more repetitions of the term
?	The preceding item is optional and matched at most once.
+	The preceding item is matched one or more times.
-	Used between start and end of a range.
\	An escape character to indicate that the following character is a match criteria and not a grouping delimiter.
>	Redirect output

Table 11: Special Characters

Options	Similar to	Description
[:upper:]	[A-Z]	uppercase letters
[:lower:]	[a-z]	lowercase letters
[:alpha:]	[A-Za-z]	upper- and lowercase letters

Table 11: Special Characters (Continued)

Options	Similar to	Description
\w	[A-Za-z_]	word characters
[:alnum:]	[A-Za-z0-9]	digits, upper- and lowercase letters
[:digit:]	[0-9]	digits
\d	[0-9]	digits
[:xdigit:]	[0-9A-Fa-f]	hexadecimal digits
[:punct:]	[.,!?:...]	punctuation
[:blank:]	[\t]	space and TAB
[:space:]	[\t\n\r\f\v]	blank characters
\s	[\t\n\r\f\v]	blank characters

Pipe/Count

SR OS supports a **pipe/count** command (...| **count**) that provides a count of the number of lines that would have otherwise been displayed. The pipe/count command is particularly useful when used in conjunction with the pipe/match command in order to count the number of output lines that match a specified pattern.

For example:

```
*A:dut-c# show service service-using vprn

===============================================================================
Services [vprn]
===============================================================================
ServiceId    Type      Adm  Opr  CustomerId Service Name
-------------------------------------------------------------------------------
1            VPRN      Down Down 1
44           VPRN      Up   Up   1
100          VPRN      Down Down 1
102          VPRN      Up   Up   1
235          VPRN      Down Down 1
1000         VPRN      Down Down 1000
-------------------------------------------------------------------------------
Matching Services : 6
-------------------------------------------------------------------------------
===============================================================================
*A:dut-c# show service service-using vprn | match Down | count
Count: 4 lines
*A:dut-c#
```

Redirection

The SR OS upports redirection (“>”) which allows the operator to store the output of a CLI command as a local or remote file. Redirection of output can be used to automatically store results of commands in files (both local and remote).

```
‘ping <customer_ip> > cf3cf1:/ping/result.txt’
‘ping <customer_ip> > ftp://ron@ftp.alcatel.com/ping/result.txt’
```

In some cases only part of the output might be applicable. The pipe/match and redirection commands can be combined:

```
ping 10.0.0.1 | match expression “time.\d+” > cf3cf1:/ping/time.txt
```

This records only the RTT portion (including the word “time”).

VI Editor

Note that “vi”sual editor (vi) is a file editor that can edit any ASCII file. This includes configuration, exec files, BOF and any other ASCII file on the system.

VT100 terminal mode is supported. However, if a different terminal mode is configured there will no noticeable negative effect.

When a configuration file is changed, a validation check is executed to see if the user is allowed to view or perform configuration changes. When a user is modifying the configuration file using the `vi` editor these checks do not occur. Because of this, the `vi` editor is only available to a user with administrator privileges. Should others require access to the `vi` editor, their profile must be modified allow the access. Access permission for the file directory where the file resides must be performed before a user can opens, read, or write a file processing command. If a user does not have permission to access the directory then the operation must be denied.

When opening a file, a resource check verifies that sufficient resources are available to process that file. If there are not enough resources, then the operation is denied and the operator is informed of that event.

Multiple sessions are allowed and are limited only by the memory resources available on the node.

Summary of vi Commands

The `vi` editor operates in two modes:

- Command mode — This mode causes actions to be taken on the file.

  In the this mode, each character entered is a command that does something to the text file being edited; a character typed in the command mode may even cause the `vi` editor to enter the insert mode.
- Insert mode — Entered text is inserted into the file.

  In the insert mode, every character typed is added to the text in the file. Hitting the `Esc` (Escape) key turns off the insert mode.

Using the vi Commands

Use the following commands to start and end **vi** edit sessions, move around in a file, enter new text, modify, move, and delete old text, as well as read from and write to files other files. Although there are numerous **vi** commands, only a few are usually sufficient to vi users. The following tables list **vi** commands.

Table 12: Cutting and Pasting/Deleting Text in vi

vi Command	Description
"	Specify a buffer to be used any of the commands using buffers. Follow the " character with a letter or a number, which corresponds to a buffer.
d	Deletes text. "dd" deletes the current line. A count deletes that many lines. Whatever is deleted is placed into the buffer specified with the " command. If no buffer is specified, then the general buffer is used.
D	Delete to the end of the line from the current cursor position.
p	Paste the specified buffer after the current cursor position or line. If no buffer is specified (with the " command.) then 'p' uses the general buffer.
P	Paste the specified buffer before the current cursor position or line. If no buffer is specified (with the " command.) then P uses the general buffer.
x	Delete character under the cursor. A count tells how many characters to delete. The characters will be deleted after the cursor.
X	Delete the character before the cursor.
y	Yank text, putting the result into a buffer. yy yanks the current line. Entering a number yanks that many lines. The buffer can be specified with the " command. If no buffer is specified, then the general buffer is used.
Y	Yank the current line into the specified buffer. If no buffer is specified, then the general buffer is used.

Table 13: Inserting New Text

vi Command	Description
A	Append at the end of the current line.
I	Insert from the beginning of a line.
O	Enter insert mode in a new line above the current cursor position.
a	Enter insert mode, the characters typed in will be inserted after the current cursor position. A count inserts all the text that was inserted that many times.
i	Enter insert mode, the characters typed in will be inserted before the current cursor position. A count inserts all the text that was inserted that many times.
o	Enter insert mode in a new line below the current cursor position.

Table 14: Moving the Cursor Within the File

vi Command	Description
^B	Scroll backwards one page. A count scrolls that many pages.
^D	Scroll forwards half a window. A count scrolls that many lines.
^F	Scroll forwards one page. A count scrolls that many pages.
^H	Move the cursor one space to the left. A count moves that many spaces.
^J	Move the cursor down one line in the same column. A count moves that many lines down.
^M	Move to the first character on the next line.
^N	Move the cursor down one line in the same column. A count moves that many lines down.
^P	Move the cursor up one line in the same column. A count moves that many lines up.
^U	Scroll backwards half a window. A count scrolls that many lines.
$	Move the cursor to the end of the current line. A count moves to the end of the following lines.
%	Move the cursor to the matching parenthesis or brace.
^	Move the cursor to the first non-whitespace character.
(	Move the cursor to the beginning of a sentence.
)	Move the cursor to the beginning of the next sentence.

Table 14: Moving the Cursor Within the File

vi Command	Description
{	Move the cursor to the preceding paragraph.
}	Move the cursor to the next paragraph.
|	Move the cursor to the column specified by the count.
+	Move the cursor to the first non-whitespace character in the next line.
-	Move the cursor to the first non-whitespace character in the previous line.
_	Move the cursor to the first non-whitespace character in the current line.
0	Move the cursor to the first column of the current line.
B	Move the cursor back one word, skipping over punctuation.
E	Move forward to the end of a word, skipping over punctuation.
G	Go to the line number specified as the count. If no count is given, then go to the end of the file.
H	Move the cursor to the first non-whitespace character on the top of the screen.
L	Move the cursor to the first non-whitespace character on the bottom of the screen.
M	Move the cursor to the first non-whitespace character on the middle of the screen.
W	Move forward to the beginning of a word, skipping over punctuation.
b	Move the cursor back one word. If the cursor is in the middle of a word, move the cursor to the first character of that word.
e	Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the last character of that word.
h	Move the cursor to the left one character position.
j	Move the cursor down one line.
k	Move the cursor up one line.
l	Move the cursor to the right one character position.
w	Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the first character of the next word.

Table 15: Moving the Cursor Around the Screen

vi Command	Description
^E	Scroll forwards one line. A count scrolls that many lines.
^Y	Scroll backwards one line. A count scrolls that many lines.
z	Redraw the screen with the following options. `z<return>` puts the current line on the top of the screen; `z.` puts the current line on the center of the screen; and `z-` puts the current line on the bottom of the screen. If you specify a count before the `z` command, it changes the current line to the line specified. For example, `16z.` puts line 16 on the center of the screen.

Table 16: Replacing Text

vi Command	Description
C	Change to the end of the line from the current cursor position.
R	Replace characters on the screen with a set of characters entered, ending with the Escape key.
S	Change an entire line.
c	Change until `cc` changes the current line. A count changes that many lines.
r	Replace one character under the cursor. Specify a count to replace a number of characters.
s	Substitute one character under the cursor, and go into insert mode. Specify a count to substitute a number of characters. A dollar sign ($) will be put at the last character to be substituted.

Table 17: Searching for Text or Characters

vi Command	Description
,	Repeat the last f, F, t or T command in the reverse direction.
/	Search the file downwards for the string specified after the /.
;	Repeat the last f, F, t or T command.
?	Search the file upwards for the string specified after the `?`.

Table 17: Searching for Text or Characters (Continued)

vi Command	Description (Continued)
F	Search the current line backwards for the character specified after the 'F' command. If found, move the cursor to the position.
N	Repeat the last search given by / or ?, except in the reverse direction.
T	Search the current line backwards for the character specified after the T command, and move to the column after the if it's found.
f	Search the current line for the character specified after the f command. If found, move the cursor to the position.
n	Repeat last search given by / or ?.
t	Search the current line for the character specified after the t command, and move to the column before the character if it's found.

Table 18: Manipulating Character/Line Formatting

vi Command	Description
~	Switch the case of the character under the cursor.
<	Shift the lines up to where to the left by one shiftwidth. << shifts the current line to the left, and can be specified with a count.
>	Shift the lines up to where to the right by one shiftwidth. >> shifts the current line to the right, and can be specified with a count.
J	Join the current line with the next one. A count joins that many lines.

Table 19: Saving and Quitting

vi Command	Description
ZZ	Exit the editor, saving if any changes were made.

Table 20: Miscellaneous

vi Command	Description
^G	Show the current filename and the status.
^L	Clear and redraw the screen.
^R	Redraw the screen removing false lines.
^[	Escape key. Cancels partially formed command.
^^	Go back to the last file edited.
!	Execute a shell. Not supported

Table 20: Miscellaneous (Continued)

vi Command	Description (Continued)
&	Repeat the previous `:s` command.
.	Repeat the last command that modified the file.
:	Begin typing an EX editor command. The command is executed once the user types return.
@	Type the command stored in the specified buffer.
U	Restore the current line to the previous state before the cursor entered the line.
m	Mark the current position with the character specified after the 'm' command.
u	Undo the last change to the file. Typing 'u' again will re-do the change.

EX Commands

The vi editor is built upon another editor, called EX. The EX editor only edits by line. From the vi editor you use the : command to start entering an EX command. This list given here is not complete, but the commands given are the more commonly used. If more than one line is to be modified by certain commands (such as :s and :w) the range must be specified before the command. For example, to substitute lines 3 through 15, the command is :3,15s/from/this/g.

Table 21: EX commands

vi Command	Description
:ab string strings	Abbreviation. If a word is typed in vi corresponding to string1, the editor automatically inserts the corresponding words. For example, the abbreviation :ab usa United States of America would insert the words, United States of America whenever the word usa is typed in.
:map keys new_seq	Mapping. This lets you map a key or a sequence of keys to another key or a sequence of keys.
:q	Quit vi. If there have been changes made, the editor will issue a warning message.
:q!	Quit vi without saving changes.
:s/pattern/ to_pattern/ options	Substitute. This substitutes the specified pattern with the string in the to_pattern. Without options, it only substitutes the first occurrence of the pattern. If a 'g' is specified, then all occurrences are substituted. For example, the command :1,$s/Alcatel/Alcatel-Lucent/g substitutes all occurrences of Alcatel to Alcatel-Lucent.
:set [all]	Sets some customizing options to vi and EX. The :set all command gives all the possible options.
:una string	Removes the abbreviation previously defined by :ab.
:unm keys	Removes the remove mapping defined by :map.
:vi filename	Starts editing a new file. If changes have not been saved, the editor will give you a warning.
:w	Write out the current file.
:w filename	Write the buffer to the filename specified.
:w >> filename	Append the contents of the buffer to the filename.
:wq	Write the buffer and quit.

Configuration Rollback

The Configuration Rollback feature provides the ability to “undo” configuration and reverts back to previous router configuration states while minimizing impacts to services.

This feature gives the operator better control and visibility over the router configurations and reduces operational risk while increasing flexibility and providing powerful recovery options.

Configuration Rollback is useful in cases where configuration changes are made but the operator later decides to not keep the changes (for example, experimentation or when problems are identified in the configuration during actual network operation).

The advantage of this feature are the following:

- Changes made to router configuration is performed with minimal impact on services being provided by the SR by not having to reboot the router.
- No impact in areas of configuration that did not change.

With the rollback feature, the operator can smoothly revert to previous configurations.

Configuration parameters that changed (or items that changed configuration have dependencies on) are first removed (revert to default), and the previous values are then restored (can be briefly service impacting in changed areas).

A history of changes is preserved (checkpoint ids) that allows rollback to different points, as well as examination of changes made as shown in Figure 3.

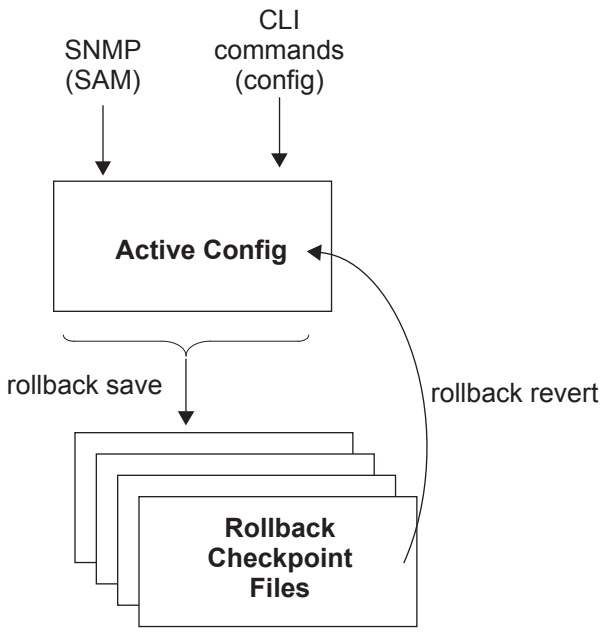

Figure 3: Rollback Operation

Feature Behavior

The following list describes detailed behavior and CLI usage of the rollback feature:

- The user can create a rollback checkpoint, and later, revert to this checkpoint with minimal impacts to services:

```
admin>rollback# save [comment <comment-string>]
comment-string: an 255 char comment associated with the checkpoint
```

- Rollback checkpoints include all current operationally active configuration:
  - Changes from direct CLI commands in the configuration branch.
  - SNMP sets
- Rollback checkpoints do not include bof configurations. The BOF file (and bof config) is not part of a rollback-save or rollback. A rollback does not change any of the bof configuration. The BOF contains basic information for the node and does not change frequently (mostly during initial commissioning of the node).
- A rollback save feature can be automatically executed (scheduled, for example, monthly) using the cron facility of SR-OS.
- The latest rollback checkpoint file uses a suffix of ".rb". The next latest rollback checkpoint file has a suffix of ".rb.1", the next oldest has a suffix of "rb.2" etc:

```
file-url.rb    <--- latest rollback file
file-url.rb.1
…
file-url.rb.9  <--- oldest rollback file
```

- When a **rollback save** is executed, the system shifts the file suffix of all the previous checkpoints by 1 (new id = old id + 1). If there are already as many checkpoint files as the maximum number supported then the last checkpoint file is deleted.
- The maximum number of rollback checkpoints is configurable and defaults to 10 ("latest" and 1 through 9, where checkpoint file 9 is deleted during the next rollback-save).
- The location and name of the rollback checkpoint files is configurable to be local (on compact flash) or remote. The *file-url* must not contain a suffix (just a path/directory + filename). The suffix for rollback checkpoint files is ".rb" and is automatically appended to rollback checkpoint files.

```
config>system>rollback# rollback-location <file-url>
```

- There is no default rollback-location. If one is not specified (or it is cleared using "no rollback-location") and a rollback save is attempted, the rollback save will fail and return an error message.
- The entire set of rollback checkpoint files can be copied from the active CPM CF to the inactive CPM CF. This synchronization is done via the following command:

```
admin>redundancy# rollback-sync
```

- The operator can enable automatic synchronization of rollback checkpoint files between the active CPM and inactive CPM. When this automatic synchronization is enabled, a rollback save will cause the new checkpoint file to be saved to both the active and standby. The suffixes of the old checkpoint files on both active and standby CPMs are incremented.

Note: The automatic sync only causes the ONE new checkpoint file to be copied to both CFs (the other 9 checkpoints are not automatically copied from active to standby but that can be done manually with admin red rollback-sync).

```
config>redundancy# [no] rollback-sync
```

- "**config red sync** {boot-env|config}" and "**admin red sync** {boot-env|config}" do not apply to rollback checkpoint files. These commands do not manually or automatically sync rollback checkpoint files. The dedicated rollback-sync commands must be used to sync rollback checkpoint files.
- Rollback files can be deleted using a dedicated rollback checkpoint deletion command.

```
admin>rollback# delete {latest-rb|<checkpoint-id>}
```

  - Deleting a rollback checkpoint causes the suffixes to be adjusted (decremented) for all checkpoints older that the one that was deleted (to close the "hole" in the list of checkpoint files and create room to create another checkpoint)
  - If "config redundancy rollback-sync" is enabled, a rollback delete will also delete the equivalent checkpoint on the standby CF and shuffle the suffixes on the standby CF.
  - If an operator manually deletes a rollback checkpoint file (using file delete) then the suffixes of the checkpoint files are NOT shuffled, nor is the equivalent checkpoint file deleted from the standby CF. This manual deletion creates a "hole" in the checkpoint file list until enough new checkpoints have been created to roll the "hole" off the end of the list.
- As shown in Figure 4, support for rolling back to a previous configuration (a saved rollback checkpoint) with minimal impact on services. The previous configuration will be loaded and take operational effect:

```
admin>rollback# revert [latest-rb|<checkpoint-id>]
```

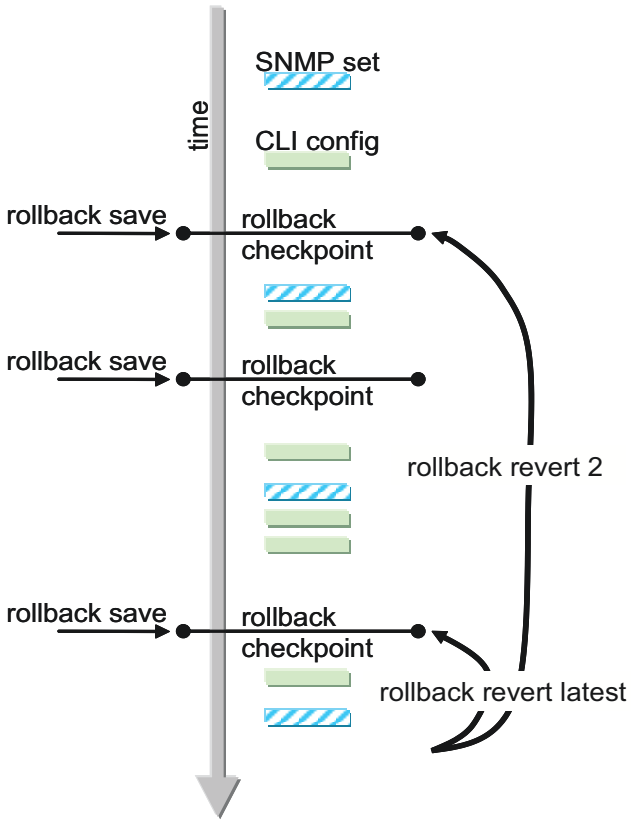

Figure 4: Configuration Rollback

- A rollback revert does not affect the currently stored rollback checkpoint files (no deletions or renumbering). This means that if an operator issues a "rollback revert 3" and then issues a "rollback-save", the resulting rollback checkpoint files "file-url.rb" and "file-url.rb.4" will contain the same rollback state/configuration.
- The **boot-good-exec** or **bad-exec** are not automatically executed after a rollback.
- impacts to the running services are minimized during a rollback:
  - no impact in areas of configuration that did not change
  - configuration parameters that changed (or items that changed config have dependencies on) are first removed (revert to default) and the previous values are then restored (can be briefly service impacting in changed areas). Some examples are the following:
    - If the currently active config contains **configure port 5/1/1 dwdm tdcm dispersion -1000** and the rollback checkpoint contains **configure port 5/1/1 dwdm tdcm dispersion -1010**, then the operational dispersion will transition from -1000, to 0 and then back to -1010 for port 5/1/1 which will cause a traffic interruption.
    - Changing the neighbor of a MC-APS port will start with neighbor 1, then be configured as no neighbor, and then will be configured with neighbor 2. Moving through the **no neighbor** intermediate state requires the working and protect circuits to be torn down and then rebuilt.

- A Rollback will undo any SNMP sets or direct CLI config commands that occurred since the last checkpoint creation.
- During the period when an SR-OS node is processing a rollback revert, both CLI commands (from other users) and SNMP commands will continue to be processed. The only commands that are blocked during a rollback revert are other rollback commands including **revert**, **save**, and **compare** (only one **rollback** command can be executing at a time in one node).
- Commands are available to view and compare the various rollback checkpoints to current operating and candidate configurations.
- Rollback checkpoint files are not guaranteed to be in any particular format. They are not interchangeable with normal config files or exec scripts. A normal config file (from an admin save) cannot be renamed as a rollback checkpoint and then referenced for a rollback revert operation. Only rollback checkpoint files generated with rollback save can be used to rollback revert.
- If a hardware change is made after a rollback-save then:
  - a rollback can be executed as long as the hardware change was an addition of hardware to the node (for example, added a new IOM into a previously empty slot).
  - a rollback is not guaranteed to work if hardware was removed or changed (for example, IOM was removed, or MDA was swapped for a different MDA type).
- Rollback across a change to the following parameters is not supported:
  - chassis-mode
  - mixed-mode
  - the SR | SS capability of a card (**configure card capability sr|ess**)
  - configure isa application-assurance-group minimum-isa-generation
- Rollback is supported even after an **admin reboot** is performed (or changes the primary config in the bof is changed and an **admin reboot** is performed). **Admin reboot** does not "break the chain" for rollback.
- The Configuration Rollback feature is incompatible with the use of Time Of Day (ToD) policies and functionality. Rollback save and rollback revert operations are blocked if any ToD policies are active (for example, assigned to objects such as a SAP).
- Lawful Intercept configuration under the **config>li** branch is not affected by a rollback or rescue. LI configuration is not saved in the rollback checkpoint or rescue file, and a rollback revert does not touch any config under the **config>li** branch.
- Any configuration or state change performed under the debug branch of CLI is not saved in the rollback checkpoint file nor impacted by a rollback.
- Rollbacks to a checkpoint created in a more recent release is not supported (for example, node running in 9.0r5 cannot rollback to a checkpoint created in 9.0r7).

- The following list captures some side effects and specific behaviors of a Rollback revert. Some of these side effects are not related purely to configuration (that is, in the CLI config branch) and may have interactions with tools commands, RADIUS, etc.
  - SAA jobs that are running when a rollback revert is initiated, and need configuration changes due to the rollback, will be stopped. If the SAA job is a continuous type then it will be re-started as part of the rollback revert after the config changes have been applied (just as if the operator had typed “no shutdown” for the continuos SAA job). Non-continuous SAA jobs that were modified by the rollback would need to be manually restarted if they need to be run again.
  - If **max-nbr-mac-addr** is reduced as part of the revert and the number of mac addresses in the forwarding database is greater than the max-nbr-mac-addr, then the rollback is aborted (before any actions are taken) and an informative error message is provided. The operator must take actions to remove the mac addresses if they wish to proceed with the rollback.
  - If active subscribers and/or subscriber hosts and/or DHCP lease state are present in the system then some associated configuration changes may be blocked (just as those same changes would be blocked if an operator tried to make them via CLI – trying to delete an sla-profile being used by active subscriber hosts, or trying to change a nat-policy in a sub-profile). If certain configuration changes associated with the hosts or lease states are required as part of the rollback but those changes are blocked, then for each blocked configuration item a warning will be printed, that particular configuration item will not be changed and the rollback will continue.
  - After multi-chassis peer shutdown or configuration changes that affect the contents of the distributed database (for example, sync tag creation or deletion), further configuration changes related to that peer may be temporarily refused. The duration of the temporary configuration freeze will depend on the size of the distributed database. A rollback attempting to make those refused configuration changes will fail and an error message will be provided to the CLI user.
  - If a **force-switchover** command (for example, **tools perform service id 1 endpoint "x" force-switchover spoke-sdp-fec 1**) has been applied to a spoke-sdp-fec of a dynamic multi-segment pseudo wire, and a rollback revert needs to change the admin state of the spoke-sdp-fec (for example, to modify spoke-sdp-fec parameters that may

be dependant on admin state), then the rollback revert will automatically remove the force-switchover and the node will revert to whatever is the best spoke-sdp in the redundant set.

- Rollback impacts the configuration state of the router, and as with normal operator CLI or SNMP configuration changes, additional actions or steps may need to occur before certain configuration changes take operational effect. Some examples include:
  - Configuration changes that require a **shutdown** and then **no-shutdown** to be done by an operator in order to take operational effect also need this manual shut/no-shut to be performed by the operator in order to take operational effect after a rollback if the rollback changes those configuration items. Some examples include:
  - Changes to Autonomous System or Confederation value require a BGP shut/no-shut.
  - Changes to VPRN Max-routes requires a shut/no-shut on the VPRN service.
  - Changes to OSPF/ISIS export-limit require a shut/no-shut on OSPF/ISIS.
  - Configuration changes to an msap-policy that normally requires a **tools perform subscriber-mgmt eval-msap** command to take operational effect on subscribers that are already active. Rollback will change the msap-policy configuration, but if it is required to have the configuration changes applied to the active subscribers then the operator will have to run the eval-msap tools command.
- Any uncommitted changes (that is, the **begin** command was entered, some changes made, but the **commit** command was never entered) in the following areas will be lost/cleared when a rollback revert is initiated:
  - **configure>application-assurance>group policy**
  - **configure>router>policy-options**
  - **configure>system>sync-if-timing**

- Some **card** and **mda** commands require a reboot, remove or rebuild of an entire card or MDA. When these commands need to be executed as part of a rollback, the impacted cards/mdas will be listed in a warning and the operator will be prompted with a single y/n prompt to decide whether to proceed or not. This prompting will not occur for a rollback initiated via SNMP, nor if the operator uses the **now** keyword with the rollback revert command. Some examples of card and mda commands that may cause a prompt are:
  - **configure>card>card-type**
  - **configure>card>named-pool-mode**
  - **configure>card>mda**
  - **configure>card>mda>mda-type**
- Although the use of the Control-C key combination is not recommended during a rollback revert, it is supported (via CLI or SNMP). Interrupting a rollback revert may leave the router in a state that is not necessarily something between the old active config and the rollback checkpoint since the rollback processing may have been in the middle of tearing things down or rebuilding configurations. A strong warning is issued in this case to indicate that the operator must examine the config and potentially issue another rollback revert to return to a known (and coherent) configuration.

- An HA CPM switchover during a rollback revert will cause the rollback operation to abort. The newly active CPM will have an indeterminate configuration. When an HA switchover occurs during a rollback (or within a few seconds of a rollback completing), the operator is advised to repeat the rollback revert operation to the same checkpoint.
- A rollback revert operation does not check authorization of each command that is applied during the revert. Permission to execute the revert operation (authorization for the “admin rollback revert” command itself) should only be given to users who are allowed to initiate a rollback revert. It is generally advised to only allow system administrators access to the file system where the rollback files are stored so that they cannot be manually edited.

Rollback and SNMP

SR OS has SNMP support for Rollback status and control. See the TIMETRA-SYSTEM-MIB for details (for example, items such as tmnxSysRollbackStarted).

When the SR OS router is doing a rollback revert, SNMP managers will see a tmnxSysRollbackStarted trap, then a rapid set of “config change” traps, and then finally, the tmnxSysRollbackStatusChange trap.

During the period when an SR OS router is processing a rollback revert, both CLI commands (from other users) and SNMP commands will continue to be processed.

Rescue Configuration

A special rescue configuration checkpoint can be created that an operator can rollback revert to at any time. The rescue configuration has its own keyword (**rescue**) and does not use the same rolling suffix indices as the normal rollback checkpoints. This allows the operator to easily return to the rescue configuration state without having to consider a checkpoint index, and ensures that the rescue checkpoint is always available (does not roll off the bottom of the list of checkpoints).

The operator should define a basic rescue configuration that is known to work and give correct management access to the node.

The location and filename of the rescue file are configurable. SR-OS appends an ".rc" suffix to the specified rescue filename.

Operational Guidelines

The following points offer some operational guidance on the usage of rollback.

- Both **admin save** and **rollback save** should be performed periodically:
- Use **admin save** to backup a complete configuration file that can be used during router reboot.
  - Used with a reboot as a last resort.
  - Do an admin save after any major h/w changes or major service changes.
  - Should be performed after any software upgrade.
- Use **rollback-save** to create a rollback checkpoint.
  - Used for intermediate checkpoints that can be recovered with minimal impacts to services.
  - Should be performed each time that a moderate amount configuration changes have been made.
  - Should be performed after any h/w changes.
  - Should be performed after any s/w upgrade.
  - Could also be scheduled with cron (for example, once every 1 or 2 weeks).
- A new **rescue-save** must be created when hardware is changed.
- Rollback-checkpoint files are not editable nor compatible/interchangeable with config files (generated with **admin save**).
- Do not continue to repeat the **rollback save**, **rollback save**, **rollback save** over the course of weeks/months without also doing executing an occasional **admin save**. In a serious situation, use one of the saved configs to use as the primary config for an **admin reboot**.
- Software Upgrade: It is recommended to create a Rollback Checkpoint (**admin rollback save**), in addition to saving the configuration (**admin save**), after an upgrade has been performed and the system is operating as expected. This will ensure a good checkpoint fully compatible with the new release is available at a point shortly after the upgrade.
- An operator could create a set of rollback checkpoints to support busy/quiet days or weekend/weekday and use cron to shift between them.
- It is beneficial to create a rollback checkpoint before a rollback revert is initiated (especially if there have been significant config changes since the last checkpoint was created). If the rollback is especially significant (a lot of major changes) it is also a good practice to do perform an **admin save** in case a full reboot is required to recover from an issue.
- A rollback failure may occur in some limited cases where the node needs a long time to complete one of the resulting configuration changes. Some examples include X and Y. If a rollback (for example, rollback revert 5) fails during execution, it should be attempted again. The second attempt will typically complete the remaining configuration changes required to fully revert to the desired checkpoint.

- When a new backup CPM is commissioned, the user execute the **admin redundancy rollback-sync** command to copy the entire set of rollback files from the active CPM cf to the new standby CPM cf. If the operator wants the system to automatically copy new rollback checkpoints to both cfs whenever a new checkpoint is created, then the **config redundancy rollback-sync** should be configured.
- An HA CPM switchover during a rollback revert will cause the rollback operation to abort. The newly active CPM will have an indeterminate configuration. A log event is created in this case to warn the operator. When an HA switchover occurs during a rollback (or within a few seconds of a rollback completing), the operator is advised to repeat the rollback revert operation to the same checkpoint.
- A rollback checkpoint stores the rollback-location and the local/remote-max-checkpoint values, and as such a rollback revert operation can change those values. If an operator changes the local/remote-max-checkpoint values it is recommended to delete all the existing checkpoints (otherwise a subsequent rollback revert could change the max back to a previous value).
- If a warning prompt (**y/n**) is displayed when a rollback revert is initiated, it is highly suggested to respond **no** to the warning prompt the first time, save a rollback checkpoint before attempting this rollback revert, and then executing the revert again and responding **yes**. If the rollback encounters problems then a revert to the saved checkpoint can be used to go back to the initial configuration state.

Transactional Configuration

Transactional configuration allows an operator to edit a candidate configuration (a set of configuration changes) without actually causing operational changes in the router (the active or operational configuration). Once the candidate configuration is complete the operator can explicitly commit the changes and cause the entire new configuration to become active.

Transactional configuration gives the operator better control and visibility over their router configurations and reduce operational risk while increasing flexibility.

Transactional Configuration and Configuration Rollback support combine to provide the operational model depicted in Figure 5.

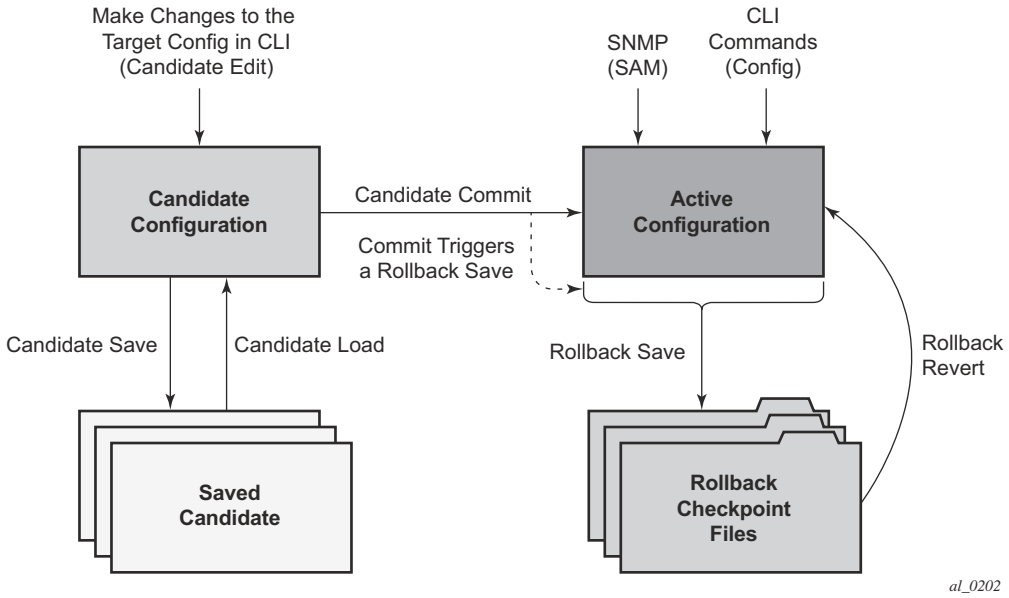

Figure 5: Router Configuration with Rollback and Transactions

Basic Operation

In order to edit the candidate configuration the operator must first enter the candidate edit mode (edit-cfg). The operator can enter and quit the configuration mode as many times as they wish before finally committing the candidate.

In edit-cfg mode the operator builds a set candidate configuration changes using the same CLI tree as standard (line-by-line non-transactional) configuration. Tab completion and keyword syntax checking is available.

Just as there is a single operational active configuration that can be modified simultaneously by multiple users in SR OS, there is also a single global candidate configuration instance. All users make changes in the same global candidate configuration and a commit operation by any user will commit the changes made by all users.

Users have the ability to exclusively create a candidate configuration by blocking other users (and sessions of the same user) from entering edit-cfg mode.

If a commit operation is successful then all of the candidate changes will take operational effect and the candidate is cleared. If there is an error in the processing of the commit, or a ‘commit confirmed’ is not confirmed and an auto-revert occurs, then the router will return to a configuration state with none of the candidate changes applied. The operator can then continue editing the candidate and try a commit later.

All commands in the candidate configuration must be in the correct order for a commit to be successful. Configuration that depends on other candidate objects must be placed after those objects in the candidate. A set of candidate editing commands (**copy**, **insert**, etc) are available to correct and reorder the candidate configuration.

The edit-cfg mode is primarily intended for building a candidate configuration while navigating the **configure** branch of CLI. Many CLI commands in branches other than **configure** are supported while in edit-cfg mode, but access to some CLI branches and command are blocked including the:

- **exec** command
- **enable-admin** command
- **enable-dynamic-services-config** command
- **admin** branch
- **bof** branch
- **debug** branch
- **tools** branch

The candidate configuration can be saved to a file and subsequently loaded into a candidate configuration. A saved candidate is similar to, but not the same as an SR OS config file generated with an **admin save** command. The saved candidate cannot be used in general as a configuration file and may not **exec** without failures.

There is no SNMP access to the candidate configuration and no SNMP management of candidates although any configuration changes done via a transaction are reported via the standard SR OS SNMP change traps and basic candidate status information is available via SNMP.

Failure of a commit may be due to one or more of several reasons including:

- Misordering: The candidate configuration has changes that are not in the correct order (an object is referred to before it is actually created).
- Invalid options and combinations: Although many syntax errors are eliminated during the candidate editing process, the candidate configuration may contain combinations of configuration and options that are not valid and are rejected when SR OS attempts to have them take operational effect.
- Out of resources: The application of the candidate may exhaust various system resources (queues, for example).

Error messages that will help the operator to take necessary actions to correct the candidate are provided for commit failures.

Standard line-by-line (immediate operational effect upon pushing the enter/return key) non-transactional CLI and SNMP commands are not blocked during the creation/editing of a candidate or the processing of a commit. These commands take immediate effect as normal.

Transactions and Rollback

By default, the SR OS will automatically create a new rollback checkpoint after a commit operation. The rollback checkpoint will include the new configuration changes made by the commit. An optional **no-checkpoint** keyword can be used to avoid the auto-creation of a rollback checkpoint after a commit. If the commit fails then no new rollback checkpoint is created.

When the **commit** confirmed option is used then a rollback checkpoint is created after the processing of the commit and will exist whether the commit is automatically reverted or not.

Transactional configuration relies on the rollback mechanism to operate. Any commands and configuration that is not supported in a rollback revert are also not supported in edit-cfg mode (examples include changes to chassis-mode or the existence of time-of-day suites).

Authorization

Authorization works transparently in edit-cfg mode and no unique/new local profile or TACACS+ permissions rules are required (other than allowing access to the **candidate** branch). For example: if an operator has permissions to access the **configure filter** context then they will automatically also have access to the **configure filter** context when in edit-cfg mode.

The candidate **load** and **save** operations (if the operator's profile allows access to the candidate load and save commands) will load and save only those items that the user is authorized to access.

The candidate view will only display the items that the user is authorized to access.

The various candidate editing commands (such as adding lines, removing lines, delete, etc) only allow operations on items that the user is authorized to access.

The candidate **commit** and **discard** operations (along with **rollback revert**) operate on the entire candidate and impact all items (authorization does not apply).