CLI Tips, Techniques, and Shortcuts

This chapter describes techniques for using the command-line interface (CLI) of the Cisco IOS XR software.

Contents

Note Commands can be entered in uppercase, lowercase, or mixed case. Only passwords are case sensitive. However, the Cisco Systems documentation convention presents commands in lowercase.

CLI Tips and Shortcuts

The following sections describe tips and shortcuts useful when using the CLI:

Entering Abbreviated Commands

You can abbreviate commands and keywords to the number of characters that allow a unique abbreviation. For example, the **configure** command can be abbreviated as **config** because the abbreviated form of the command is unique. The router accepts and executes the abbreviated command.

Using the Question Mark (?) to Display On-Screen Command Help

Use the question mark (**?**) to learn what commands are available and the correct syntax for a command. Table 23 summarizes the options for on-screen help.

Tip The space (or no space) before the question mark (**?**) is significant. If you include a space before the question mark, the system displays all available options for a command or CLI mode. If you do not include a space, the system displays a list of commands that begin with a particular character string.

Table 23 On-Screen Help Commands

Command	Description

partial-command **?**	Enter a question mark (**?**) at the end of a partial command to list the commands that begin with those characters.
	`RP/0//CPU0:router# co?`
	`configure copy`
	Note Do not include a space between the command and question mark.

?	Lists all commands available for a particular command mode.

command **?**	Include a space before the question mark (**?**) to list the keywords and arguments that belong to a command.
	`RP/0//CPU0:router# configure ?`
	`exclusive Configure exclusively from this terminal`
	`terminal Configure from the terminal`
	`<cr>`
	Note For most commands, the <cr> symbol indicates that you can execute the command with the syntax already entered. For the preceding example, press **Return** to enter global configuration mode.

command keyword **?**	Enter a question mark (?) after the keyword to list the next available syntax option for the command.
	`RP/0//CPU0:router# show aaa ?`
	`ikegroup Show local IKE group(s)`
	`locald locald sub system`
	`login login sub system`
	`task Show task information`
	`taskgroup Show all the local taskgroups configured in the system`
	`trace Show trace data for AAA sub system`
	`userdb Show all local users with the usergroups each belong to`
	`usergroup Show all the local usergroups configured in the system`
	Note Include a space between the keyword and question mark.

The following example shows how to add an entry to access list 99. The added entry denies access to all hosts on subnet 172.0.0.0 and ignores bits for IPv4 addresses that start within the range of 0 to 255. The following steps provide an example of on-screen command help:

Step 1 Enter the **access-list** command, followed by a space and a question mark, to list the available options for the command:

```
RP/0//CPU0:router(config)# ipv4 access-list ?

  log-update  Control access lists log updates
  ssm-acl     Access list name - maximum 32 characters
  bidir-acl   Access list name - maximum 32 characters
  WORD        Access list name - maximum 32 characters
```

Note The number ranges (within the angle brackets) are inclusive ranges.

Step 2 Enter the access list name **list1**, followed by a space and another question mark, to display the arguments that apply to the keyword and brief explanations:

```
RP/0//CPU0:router(config)# ipv4 access-list list1 ?

  log-update  Control access lists log updates
  ssm-acl     Access list name - maximum 32 characters
  bidir-acl   Access list name - maximum 32 characters
  WORD        Access list name - maximum 32 characters
RP/0/RP0/CPU0:router(config)#ipv4 access-list list1 ?
  <1-2147483646>  Sequence number for this entry
  deny            Specifies packets to reject
  permit          Specifies packets to forward
  remark          Comment for access list
  <cr>
RP/0//CPU0:router(config)#ipv4 access-list list1
```

Step 3 Enter the **deny option** and a question mark to see more command options:

```
RP/0//CPU0:router(config)#ipv4 access-list list1 deny ?

  <0-255>         An IPv4 Protocol Number
  A.B.C.D         Source IP address or prefix
  A.B.C.D/prefix  Source IP address and care bits
  ahp             Authentication Header Protocol
  any             Any source host
  eigrp           Cisco's EIGRP Routing Protocol
  esp             Encapsulation Security Payload
  gre             Cisco's GRE Tunneling
  host            A single source host
  icmp            Internet Control Message Protocol
  igmp            Internet Gateway Message Protocol
  igrp            Cisco's IGRP Routing Protocol
  ipinip          IP in IP tunneling
  ipv4            Any IPv4 Protocol
  nos             KA9Q NOS Compatible IP over IP Tunneling
  ospf            OSPF Routing Protocol
  pcp             Payload Compression Protocol
  pim             Protocol Independent Multicast
  sctp            Stream Control Transmission Protocol
  tcp             Transport Control Protocol
  udp             User Datagram Protocol
RP/0//CPU0:router(config)#ipv4 access-list list1 deny
```

Step 4 Enter an IP address, followed by a space and a question mark (?), to list additional options:

```
RP/0//CPU0:router(config)# ipv4 access-list list1 deny 172.31.134.0 ?

   A.B.C.D    Wildcard bits
  log        Log matches against this entry
  log-input  Log matches against this entry, including input interface
  <cr>

RP/0//CPU0:router(config)# ipv4 access-list list1 deny 172.31.134.0
```

The <cr> symbol by itself indicates that there are no more keywords or arguments.

Step 5 Press **Enter** to execute the command:

```
RP/0//CPU0:router(config)# ipv4 access-list list1 deny 172.31.134.0
```

Note The configuration does not become active until you enter the **commit** command to add the target configuration to the running configuration.

Completing a Partial Command with the Tab Key

If you do not remember a complete command name or want to reduce the amount of typing you have to perform, enter the first few letters of the command, then press the **Tab** key. If only one command begins with that character string, the system automatically completes the command for you. If the characters you entered indicate more than one command, the system beeps to indicate that the text string is not unique and the system provides a list of commands that match the text entered.

In the following example, the CLI recognizes **conf** as a unique string in EXEC mode and completes the command when you press the **Tab** key:

```
RP/0//CPU0:router# conf<Tab>
RP/0//CPU0:router# configure
```

The CLI displays the full command name. You must then press **Return** to execute the command. This feature allows you to modify or reject the suggested command.

In the next example, the CLI recognizes two commands that match the text entered:

```
RP/0//CPU0:router# co<Tab>
configure  copy
RP/0//CPU0:router# con<Tab>
RP/0//CPU0:router# configure
```

Tip If your keyboard does not have a Tab key, press **Ctrl-I** instead.

Identifying Command Syntax Errors

If an incorrect command is entered, an error message is returned with the caret (^) at the point of the error. In the following example, the caret appears where the character was typed incorrectly in the command:

```
RP/0//CPU0:router# configure termiMal
                                  ^
% Invalid input detected at '^' marker.
```

Note The percent sign (%) indicates the line in which the error message occurred.

To display the correct command syntax, enter the **?** after the command:

```
RP/0//CPU0:router# configure ?

  exclusive  Configure exclusively from this terminal
  terminal   Configure from the terminal
  <cr>
```

Using the no Form of a Command

Almost every configuration command has a **no** form. Depending on the command, the **no** form enables or disables a feature. For example, when configuring an interface, the **no shutdown** command brings up the interface, and the **shutdown** command shuts down the interface. The **username** command creates a new user, and the **no username** command deletes a user when entered with a valid username.

The Cisco IOS XR software command reference publications provide the complete syntax for the configuration commands and describe what the **no** form of a command does. For more information, see the "Related Documents" section on page x.

Editing Command Lines that Wrap

The CLI provides a wraparound feature for commands that extend beyond a single line on the screen. When the cursor reaches the right margin, the command line shifts ten spaces to the left. The first ten characters of the line are not shown, but it is possible to scroll back and check the syntax at the beginning of the command. To scroll back, press **Ctrl-B** or the **Left Arrow** key repeatedly, or press **Ctrl-A** to return directly to the beginning of the line.

In the following example, the **ipv4 access-list** command entry is too long to display on one line. When the cursor reaches the end of the line, the line is shifted to the left and redisplayed. The dollar sign ($) after the command prompt indicates that the line has been scrolled to the left and the beginning of the command is hidden.

```
RP/0//CPU0:router(config)# $s-list 101 permit tcp 172.31.134.5 255.255.255.0 172.31.135.0
```

In the next example, Ctrl-A is used to display the beginning of the command line, and the dollar sign at the end of the command line shows the command has been scrolled to the right and the end of the command is hidden:

```
RP/0//CPU0:router(config)# ipv4 access-list 101 permit tcp 172.31.134.5 255.255.255.0 17$
```

In the next example, the Right Arrow key has been used to scroll to the right. Notice that dollar sign symbols appear at both ends of the line, which indicates that command information is hidden from the beginning and end of the command.

```
RP/0//CPU0:router(config)# $ccess-list 101 permit tcp 172.31.134.5 255.255.255.0 172.31.$
```

By default, the Cisco IOS XR software uses a terminal screen 80 columns wide. To adjust for a different screen width, use the **terminal width** command in EXEC mode.

Use line wrapping with the command history feature to recall and modify previous complex command entries.

Displaying System Information with show Commands

The **show** commands display information about the system and its configuration. The following sections describe some common **show** commands and provide techniques to manage the output from those commands:

Common show Commands

Table 24 shows some of the most common **show** commands.

Table 24 Common show Commands in Cisco IOS XR Software

Command	Description	Command Mode
show version	Displays system information.	EXEC or administration EXEC mode
show configuration	Displays the uncommitted configuration changes made during a configuration session.	Global or administration configuration mode
show running-config (EXEC or global configuration mode)	Displays the current running configuration for the to which you are connected.	EXEC or global configuration mode
show running-config (administration EXEC or administration configuration mode)	Displays the current running configuration that applies to the entire router	administration EXEC or administration configuration mode
show tech-support	Collects a large amount of system information for troubleshooting. You can provide this output to technical support representatives when reporting a problem.	EXEC or administration EXEC mode

Table 24 Common show Commands in Cisco IOS XR Software (continued)

Command	Description	Command Mode
show platform (EXEC mode)	Displays information about cards and modules assigned to the to which you are connected.	EXEC mode
show platform (administration EXEC mode)	Displays information about all cards and modules in the router.	administration EXEC mode
show environment	Displays hardware information for the system, including fans, LEDs, power supply voltage and current, and temperatures. Enter **show environment ?** to see additional command options.	EXEC mode or administration EXEC mode

For more information on the use of these commands, see the "Related Documents" section on page x.

Browsing Display Output When the --More-- Prompt Appears

When command output requires more than one screen, such as for the **?**, **show**, or **more** commands, the output is presented one screen at a time, and a `--More--` prompt appears at the bottom of the screen.

To display additional command output, do one of the following:

- Press **Return** to display the next line.
- Press **Spacebar** to display the next screen of output.

The following example shows one screen of data and the `--More--` prompt:

```
RP/0//CPU0:router# show ?
```

Tip If you do not see the `--More--` prompt, try entering a value for the screen length with the **terminal length** command in EXEC mode. Command output is not paused if the **length** value is set to zero. The following example shows how to set the terminal length:

```
RP/0//CPU0:router# terminal length 20
```

For information on searching or filtering CLI output, see the "Filtering show Command Output" section on page 117.

Halting the Display of Screen Output

To interrupt screen output and terminate a display, press **Ctrl-C**, as shown in the following example:

```
RP/0//CPU0:router# show running-config
<Ctrl-C>
```

Redirecting Output to a File

By default, CLI command output appears on the screen. CLI command output can be redirected to a user-specified file by entering a filename and location after the **show** command syntax. The following command syntax is used to redirect output to a file:

```
show command | file filename
```

This feature enables you to save any of the **show** command output in a file for further analysis and reference. When you choose to redirect command output, consider the following guidelines:

- If the full path of the file is not specified, the default directory for your account is used. You should always save your target configuration files to this location.
- If the saved output is to be used as a configuration file, the filename should end with the `cfg` suffix for easy identification. This suffix is not required, but can help locate target configuration files. Example: `myconfig.cfg`

In the following example, a target configuration file is saved to the default user directory:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# show configure | file disk0:myconfig.cfg
RP/0//CPU0:router(config)# abort
RP/0//CPU0:router#
```

Narrowing Output from Large Configurations

Displaying a large running configuration can produce thousands of lines of output. To limit the output of a **show** command to only the items you want to view, use the procedures in the following sections:

- Limiting show Command Output to a Specific Feature or Interface, page 116
- Using Wildcards to Display All Instances of an Interface, page 116

Limiting show Command Output to a Specific Feature or Interface

Entering keywords and arguments in the **show** command limits the **show** output to a specific feature or interface.

In the following example, only information about the static IP route configuration appears:

```
RP/0//CPU0:router# show running-config router static

router static
 address-family ipv4 unicast
  0.0.0.0/0 10.21.0.1
  0.0.0.0/0 /1/0/1 10.21.0.1
 !
!
```

In the following example, the configuration for a specific interface appears:

```
RP/0//CPU0:router# show running-config interface 0/1/0/1

interface /1/0/1
 ipv4 address 10.21.54.31 255.255.0.0
 !
```

Using Wildcards to Display All Instances of an Interface

To display the configuration for all instances, enter the asterisk (*) wildcard character.

Note For more information, see the "Using Wildcards to Identify Interfaces in show Commands" section on page 124.

In the following example, a configuration for all -interfaces is displayed:

```
RP/0//CPU0:router# show running-config interface *

interface /1/0/0
 ipv4 address 10.2.3.4 255.255.255.0

  crc 32
 !
 shutdown
 keepalive disable
!
interface /1/0/1
 ipv4 address 10.2.3.5 255.255.255.0

  crc 32
 !
 shutdown
 keepalive disable
!
interface /1/0/2
 ipv4 address 10.2.3.6 255.255.255.0

  crc 32
 !
 shutdown
 keepalive disable
!
interface /1/0/3
 ipv4 address 10.2.3.7 255.255.255.0

  crc 32
 !
 shutdown
 keepalive disable
!

 --More--
```

Filtering show Command Output

Output from the **show** commands can generate a large amount of data. To display only a subset of information, enter the "pipe" character (|) followed by a keyword (**begin**, **include**, **exclude**, or **file**) and a regular expression. Table 25 shows the filtering options for the **show** command.

Table 25 show Command Filter Options

Command	Description
show *command* | **begin** *regular-expression*	Begins unfiltered output of the **show** command with the first line that contains the regular expression.
show *command* | **exclude** *regular-expression*	Displays output lines that do not contain the regular expression.
show *command* | **include** *regular-expression*	Displays output lines that contain the regular expression.

Table 25 show Command Filter Options

Command	Description
show *command* | **file** *device0:path/file*	Saves output of the **show** command to the specified file on the specified device.
show *command* | **utility** *name*	Displays a set of UNIX utilities: • **cut**—Cuts characters or lines from the output displayed from standard input or a file. • **egrep**—Searches a file using full regular expressions. • **fgrep**—Searches a file for a fixed character string. • **head**—Copies bytes or lines at the beginning of the output displayed from standard input or a file. • **less**—Displays the output of a file in a page-by-page manner. • **sort**—Sorts, merges, or sequence-checks the output displayed from standard input or a file. • **tail**—Copies the end portion of the output displayed from standard input or a file. • **uniq**—Displays or removes repeated lines in a file. • **wc**—Count words, lines, or bytes in a file. • **xargs**—Invokes a program from one or more argument lists.

In the following example, the **show interface** command includes only lines in which the expression "protocol" appears:

```
RP/0//CPU0:router# show interface | include protocol

Null0 is up, line protocol is up
0 drops for unrecognized upper-level protocol
/2/0/0 is administratively down, line protocol is administratively down
0 drops for unrecognized upper-level protocol
/2/0/1 is administratively down, line protocol is administratively down
0 drops for unrecognized upper-level protocol
/2/0/2 is administratively down, line protocol is administratively down
0 drops for unrecognized upper-level protocol
/2/0/3 is administratively down, line protocol is administratively down
0 drops for unrecognized upper-level protocol
MgmtEthernet0//CPU0/0 is administratively down, line protocol is administratively
down
MgmtEthernet0//CPU0/0 is administratively down, line protocol is administratively
down
0 drops for unrecognized upper-level protocol
```

Note Filtering is available for submodes, complete commands, and anywhere that <cr> appears in the "**?**" output.

Adding a Filter at the --More-- Prompt

You can specify a filter at the `--More--` prompt of a **show** command output by entering a forward slash (**/**) followed by a regular expression. The filter remains active until the command output finishes or is interrupted (using **Ctrl-Z** or **Ctrl-C**). The following rules apply to this technique:

- If a filter is specified at the original command or previous `--More--` prompt, a second filter cannot be applied.
- The use of the **begin** keyword does not constitute a filter.
- The minus sign (–) preceding a regular expression displays output lines that do not contain the regular expression.
- The plus sign (+) preceding a regular expression displays output lines that contain the regular expression.

In the following example, the user adds a filter at the `--More--` prompt to show only the lines in the remaining output that contain the regular expression “ip”:

```
RP/0//CPU0:router# show configuration running | begin line

Building configuration...
line console
 exec-timeout 120 120
!
logging trap
--More--
/ip
filtering...
ip route 0.0.0.0 255.255.0.0 /2/0/0
interface /2/0/0
 ip address 172.19.73.215 255.255.0.0
end
```

Tip On most systems, **Ctrl-Z** can be entered at any time to interrupt the output and return to EXEC mode.

For more information, see Appendix A, “Understanding Regular Expressions, Special Characters, and Patterns.”

Multipipe Support

The multipipe feature supports the multiple pipes on the CLI. With this feature, the output can be processed by an enhanced utility set. Using various combination of utilities, it is possible to gather, filter, and format the output of any **show** command. An arbitrary limit of eight pipes is supported on CLI with this limit superseded by the limit of characters that can be typed on the single line (1024) if the individual commands specified with pipes are long enough.

In addition, if you want to give pipe character (|) as a pattern, you must give it in double quotes. For example:

```
RP/0//CPU0:single8-hfr# show running-config|include "|ospf"|file disk0:/usr/a.log
```

show parser dump command

The **show parser dump** command displays the CLI syntax options for a specific submode.

It is a utility that dumps the parser commands supported on the router and a tool that displays line-by-line commands available in a submode. The command is available in every mode and it shows the command set available for that mode. This is a very handy tool for collecting the CLI commands for a mode.

The **show parser dump** command supports a filter. For example, an initial portion of the command can be specified and the command set matching to that portion can be displayed.

```
RP/0//CPU0:router(config-un)# show parser dump

show
show configuration merge
show configuration running sanitized desanitize rpl
show configuration running sanitized
show configuration running
show configuration
show configuration failed noerrors
show configuration failed
show configuration failed load
show running-config
show running-config sanitized desanitize rpl
show running-config sanitized
show running-config submode
show parser dump
show history detail
show history
pwd
exit
```

Accessing Admin Commands from Secure Domain Router Mode

You can access admin commands from secure domain router mode by prefixing the **Admin** keyword. Switching to admin mode is not required. For example:

```
RP/0//CPU0:router# admin install add
tftp://223.255.254.254/muck/username/38ws/hfr-mpls-p.pie sync active
```

In the preceding example the **install** command is an admin mode command that you can run from by prefixing **admin** keyword.

Location Keyword for the file Command

Specify the location of the media (as specified, disk0) where the file needs to be stored. This option is available only for the disk or any media storage available on different nodes of the router.

If you have a media (disk0: disk1:), it is provided with an additional location keyword. This option displays all the nodes where the media is present.

```
RP/0//CPU0:router# sh logging | file disk0:/log-file location ?

0/0/cpu0 Fully qualified location specification
0/1/cpu0 Fully qualified location specification
```

Note The **location** keyword must be available only for the disk or any media storage available on RP. Network files do not require this keyword.

vty / Console Timestamp

This feature enables the timestamp to be set to *On* by default for each EXEC or admin EXEC command. Previously, the default setting for the time stamp was disabled.

The following command disables the timestamp:

```
RP/0//CPU0:router(config)# line console timestamp disable
```

The following command enables the timestamp:

```
RP/0//CPU0:router(config)# no line console timestamp disable
```

However, the previous command to enable the timestamp is still available.

Displaying Interfaces by Slot Order

This feature lets you display physical interfaces in a sequence of slots for a specific rack. This provides an easy way to determine if the interfaces are configured on a specific slot. Previously, the physical interfaces were displayed by interface types.

To display the interfaces by slot order, you need to configure the **configuration display interface slot-order** command at the global configuration mode:

```
RP/0//CPU0:router# configure terminal
RP/0//CPU0:router(config)# configuration display interface slot-order
RP/0//CPU0:router(config)# commit
RP/0//CPU0:router(config)# end
```

This command enables the display of physical interfaces by slot-order:

```
RP/0//CPU0:router# show running-config
service configuration display slot-order
interface MgmtEth0/0/CPU0/0
 ipv4 address 12.29.38.6 255.255.0.0
!
interface MgmtEth0/0/CPU0/1
 shutdown
!

interface POS0/2/0/0
 shutdown
!
interface POS0/2/0/1
 shutdown
!
interface GigabitEthernet0/3/0/0
 shutdown
!
interface GigabitEthernet0/3/0/1
 shutdown
```

```
!

interface POS0/4/0/0
 shutdown
!
interface POS0/4/0/1
 shutdown
```

Note The **configuration display interface slot-order** command is supported only in the configuration mode.

Displaying Unconfigured Interfaces

This feature lets you display the list of all physical interfaces, even if these interfaces are not configured. You can use the **show running-config all-interfaces** command to display all unconfigured interfaces. Previously, the **show running-config** command displayed only the running configuration of the system--any feature not configured explicitly by the user (or operating in default mode) would not have any evidence in the output of the **show running-config** command.

```
RP/0//CPU0:router# show running-config all-interfaces
Sun Jun 13 21:44:46.769 DST
hostname Router
interface MgmtEth0/0/CPU0/0
 ipv4 address 12.29.38.6 255.255.0.0
!
interface MgmtEth0/0/CPU0/1
 shutdown
!
interface POS0/2/0/0
!
interface POS0/2/0/1
!
router static
 address-family ipv4 unicast
   0.0.0.0/0 12.29.0.1
 !
!
```

Notice that the POS interfaces have no configurations but they are still shown in the output of the command.

This option is not applicable to other variants of show configuration commands like the following:

- **show configuration**
- **show configuration commit changes**
- **show configuration rollback changes**
- **show configuration failed**
- **show configuration persistent**

Displaying Subnet Mask in CIDR Format

This feature displays IPv4 address subnet mask in Classless Interdomain Routing (CIDR) format instead of decimal format. The change of format for all show commands may cause backward compatibility issues. To overcome this problem, the **ipv4 netmask-format hit-count** command has been implemented in the IP/CLI component, which maintains the common infrastructure specific to IP related CLIs.

To display the subnet in a prefix length format, you need to configure the **ipv4 netmask-format hit-count** command at the global configuration mode:

```
RP/0//CPU0:router# configure terminal
RP/0//CPU0:router(config)# ipv4 netmask-format bit-count
RP/0//CPU0:router(config)# commit
RP/0//CPU0:router(config)# end
```

After this command has been configured, the output of the show command forcefully displays the subnet mask in a prefix length format. Also, you can disable the command by using the **no** form of the command.

```
RP/0//CPU0:router# no ipv4 netmask-format bit-count
RP/0//CPU0:router#
```

Note This **ipv4 netmask-format hit-count** command is supported only in the configuration mode.

The following example shows the output of a **show running-config** command after the **ipv4 netmask-format bit-count** command has been configured:

```
RP/0//CPU0:router# show running-config interface mgmtEth 0/RP0/CPU0/0
Mon May 31 23:48:17.453 DST
interface MgmtEth0/RP0/CPU0/0
 description Connected to Lab LAN
 ipv4 address 172.29.52.70 255.255.255.0
```

Wildcards, Templates, and Aliases

This section contains the following topics:

Using Wildcards to Identify Interfaces in show Commands

Wildcards (*) identify a group of interfaces in **show** commands. Table 26 provides examples of wildcard usage to identify a group of interfaces.

Table 26 Examples of Wildcard Usage

Wildcard Syntax	Description
*	Specifies all interfaces
*	Specifies all interfaces in the system
/1/*	Specifies all interfaces in rack 0, slot 1
/3/4*	Specifies all subinterfaces for /3/4

Note The wildcard (*) must be the last character in the interface name.

Examples

The following example shows how the configuration for all interfaces in rack 0, slot 1 is displayed.

```
RP/0//CPU0:router:router# show running-config interface /1/*

interface /1/0/0
 ipv4 address 10.2.3.4 255.255.255.0

  crc 32
 !
 keepalive disable
interface /1/0/1
 ipv4 address 10.2.3.5 255.255.255.0

  crc 32
 !
 keepalive disable
interface /1/0/2
 ipv4 address 10.2.3.6 255.255.255.0

  crc 32
 !
 keepalive disable
interface /1/0/3
 ipv4 address 10.2.3.7 255.255.255.0
```

```
 crc 32
!
keepalive disable

--More--
```

The following example shows how the state of all interfaces is displayed:

```
RP/0//CPU0:router# show interfaces * brief

               Intf        Intf         LineP              Encap  MTU       BW
               Name        State        State               Type (byte)    (Kbps)
--------------------------------------------------------------------------------
          /1/0/0           up           up                HDLC  4474   2488320
          /1/0/1           up           up                HDLC  4474   2488320
          /1/0/2           up           up                HDLC  4474   2488320
          /1/0/3           up           up                HDLC  4474   2488320
          /1/0/4           up           up                HDLC  4474   2488320
          /1/0/5           up           up                HDLC  4474   2488320
          /1/0/6           up           up                HDLC  4474   2488320
          /1/0/7           up           up                HDLC  4474   2488320
          /1/0/8           up           up                HDLC  4474   2488320
          /1/0/9           up           up                HDLC  4474   2488320
         /1/0/10           up           up                HDLC  4474   2488320
         /1/0/11           up           up                HDLC  4474   2488320
         /1/0/12           up           up                HDLC  4474   2488320
         /1/0/13           up           up                HDLC  4474   2488320
         /1/0/14           up           up                HDLC  4474   2488320
         /1/0/15           up           up                HDLC  4474   2488320
```

Creating Configuration Templates

Configuration templates allow you to create a name that represents a group of configuration commands. After a template is defined, it can be applied to interfaces by you or other users. As networks scale to large numbers of nodes and ports, the ability to configure multiple ports quickly using templates can greatly reduce the time it takes to configure interfaces.

The two primary steps in working with templates are creating templates and applying templates. The following procedure describes how to create a configuration template.

SUMMARY STEPS

1. **configure**
2. **template** *template-name* [(**$***parameter* **$***parameter...*)] [*config-commands*]
3. Enter the template commands.
4. **end-template**
5. **commit**
6. **show running-config template** *template-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `Router# configure`	Enters global configuration mode.
Step 2	**template** *template-name* [(**$***parameter* **$***parameter...*)] [*config-commands*] **Example:** `RP/0//CPU0:router(config)# template tmplt_1`	Enters template configuration mode and creates a template. • *template-name*—Unique name for the template to be applied to the running configuration. • (Optional) *parameter*—Actual values of the variables specified in the template definition. Up to five parameters can be specified within parentheses. Each parameter must begin with the **$** character. Templates can be created with or without parameters. • (Optional) *config-commands*—Global configuration commands to be added to the template definition. Any name in a command (such as the server name, group name, and so on) can be parameterized. This means that those parameters can be used in the template commands (starting with $) and replaced with real arguments when applied. • To remove the template, use the **no** form of this command.
Step 3	Enter the template commands. **Example:** `RP/0//CPU0:router(config-TPL)# hostname test`	Defines the template commands.
Step 4	**end-template** **Example:** `RP/0//CPU0:router(config-TPL)# end-template`	Ends the template definition session and exits template configuration mode. • When you end the template session, you are returned to global configuration mode.
Step 5	**commit** **Example:** `RP/0//CPU0:router(config-TPL)# commit`	Applies the target configuration commands to the running configuration.
Step 6	**show running-config template** *template-name* **Example:** `RP/0//CPU0:router# show running-config template tmplt_1`	Displays the details of the template.

Examples

The following example shows how a simple template is defined. The template contents are then displayed with the **show running-config template** *template-name* command:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# template jbtest
RP/0//CPU0:router(config-TPL)# hostname test
RP/0//CPU0:router(config-TPL)# end-template
RP/0//CPU0:router(config)# commit
RP/0//CPU0:router(config)# show running-config template jbtest

template jbtest
 hostname test
end-template
```

In the next example, a template is defined, and the template requires a parameter. The template contents are then displayed with the **show running-config template** *template-name* command:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# template test2 (hostname)
RP/0//CPU0:router(config-TPL)# hostname $hostname
RP/0//CPU0:router(config-TPL)# end-template
RP/0//CPU0:router(config)# commit
RP/0//CPU0:router(config)# show running-config template test2

template test2 (hostname)
 hostname $hostname
end-template
```

Applying Configuration Templates

To apply a template, enter the **apply-template** *template-name* [(*parameter*)] command in global configuration mode and consider the following guidelines:

- Only one template can be applied at a time.
- If the same template is applied multiple times, the most recent application overwrites the previous ones.
- Provide the exact number of parameters for the template.
- Templates are applied as a “best effort” operation; only valid changes are committed. If any command in the template fails, that command is discarded.
- After a template is applied, the **show configuration** command displays the target configuration changes. The target configuration must be committed (with the **commit** command) to become part of the running configuration.

Examples

In the following example, a simple template is defined. The template contents are then displayed with the **show running-config template** *template-name* command:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# apply-template jbtest
RP/0//CPU0:router(config)# show
Building configuration...
hostname test
end
```

In the next example, a template with one parameter is applied and the **show configuration** command displays the result:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# apply-template test2 (router)
RP/0//CPU0:router(config)# show configuration

Building configuration...
hostname router
end
```

Aliases

With the Cisco IOS XR software, you can define command-line aliases for any physical or logical entity in a router. After you define the alias, it is used in the CLI to reference the real entity.

To create a command alias, enter the **alias** command in global configuration or administration configuration mode:

alias *alias-name* [(*parameter1 parameter2...*)] *command-syntax* [$*parameter1*] [*command-syntax*] [$*parameter2*]

Table 27 defines the **alias** command syntax.

Table 27 alias Command Syntax

Syntax	Specifies that the Alias Is Created for
alias-name	Name of the command alias. An alias name can be a single word or multiple words joined by a dash (–) delimiter.
command-syntax	Original command syntax. Valid abbreviations of the original command syntax can be entered for the *command-syntax* argument.
(*parameterx*)	Argument or keyword that belongs to the command you specified for the *command-syntax* argument. When the parameter is entered in parenthesis after the alias name, the alias requires a parameter name. To associate the parameter with a command within the alias, enter the $ character preceding the parameter name.

Multiple commands can be supported under a single command alias, and multiple variables can be supported for each command. If multiple commands are specified under a single alias, each command is executed in the order in which it is listed in the **alias** command.

In the following example, an alias named *my-cookie* is created for the Management Ethernet interface, and then the new alias is specified to enter interface configuration mode:

```
RP/0//CPU0:router(config)# alias my-cookie mgmtEth 0/0/CPU0/0
RP/0//CPU0:router(config)# interface my-cookie
RP/0//CPU0:router(config)# interface mgmtEth 0/0/CPU0/0
RP/0//CPU0:router(config-if)#
```

After you enter a command with an alias, the router displays the command you entered with the alias value so that you can verify that alias value.

To delete a specific alias, enter the **no** form of the **alias** command with the alias name.

Keystrokes Used as Command Aliases

The system can be configured to recognize particular keystrokes (key combination or sequence) as command aliases. In other words, a keystroke can be set as a shortcut for executing a command. To enable the system to interpret a keystroke as a command, use the **Ctrl-V** or **Esc**, **Q** key combination before entering the command sequence.

Command History

The Cisco IOS XR software lets you display a history of the most recently entered and deleted commands. You can also redisplay the command line while a console message is being shown. The following sections describe the command history functionality:

- Displaying Previously Entered Commands, page 129
- Recalling Previously Entered Commands, page 129
- Recalling Deleted Entries, page 130
- Redisplaying the Command Line, page 130
- Displaying Persistent CLI History, page 130

Note To roll back to a previously committed configuration, see Managing Configuration History and Rollback.

Displaying Previously Entered Commands

The Cisco IOS XR software records the ten most recent commands issued from the command line in its history buffer. This feature is particularly useful for recalling long or complex commands or entries, including access lists.

To display commands from the history buffer, enter the **show history** command as follows:

```
RP/0//CPU0:router# show history

  show configuration history commit
  show configuration commit list
show config commit changes 1000000001
  show history
```

Recalling Previously Entered Commands

The Cisco IOS XR software records the ten most recent commands issued from the command line in its history buffer. This feature is particularly useful for recalling long or complex commands or entries, including access lists.

Table 28 lists the commands or key strokes to use to recall commands from the history buffer.

Table 28 Command History

Command or Key Combination	Purpose
Ctrl-P or the **Up Arrow** key	Recalls commands in the history buffer, beginning with the most recent command. Repeat the key sequence to recall successively older commands.
Ctrl-N or the **Down Arrow** key	Returns to more recent commands in the history buffer after recalling commands with **Ctrl-P** or the **Up Arrow** key. Repeat the key sequence to recall successively more recent commands.

Recalling Deleted Entries

The Cisco IOS XR CLI also stores deleted commands or keywords in a history buffer. The buffer stores the last ten items that have been deleted using **Ctrl-K**, **Ctrl-U**, or **Ctrl-X**. Individual characters deleted using **Backspace** or **Ctrl-D** are not stored.

Table 29 identifies the keystroke combinations used to recall deleted entries to the command line.

Table 29 Keystroke Combinations to Recall Deleted Entries

Command or Key Combination	Recalls
Ctrl-Y	Most recent entry in the buffer (press the keys simultaneously).
Esc, **y**	Previous entry in the history buffer (press the keys sequentially).

Note The **Esc**, **y** key sequence does not function unless the **Ctrl-Y** key combination is pressed first. If the **Esc, y** is pressed more than ten times, the history cycles back to the most recent entry in the buffer.

Redisplaying the Command Line

If the system sends a message to the screen while a command is being entered, the current command-line entry can be redisplayed using the **Ctrl-L** or **Ctrl-R** key combination.

Displaying Persistent CLI History

The Cisco IOS XR maintains the history buffer of CLI commands persistently across user sessions, router switchover, and router reloads. This buffer not only provides a log of commands entered by various users, but also lets you trace the activity of active users if the threshold limit of CPU usage is exceeded. This command is useful for troubleshooting purposes.

To display the history of events corresponding to the CLI session open events, enter the **show cli history brief location** command at the EXEC mode as follows:

```
RP/0//CPU0:router# show cli history brief location 0/RP0/CPU0

No.        Username            Line            IPAddress     Client            t
--- --------------- --------------- -------------------- ---------- ------------
  1               -               -                    -          -  Thu Jun 11e
```

```
  2               -               -                    -          -  Thu Jun 11e
  3               -               -                    -          -  Thu Jun 11e
  4         jhensper   con0_RP0_CPU0                   -       exec  Thu Jun 11n
  5         jhensper   con0_RP0_CPU0                   -  adminexec  Thu Jun 11n
```

To display the history of commands from each session along with user name, enter the **show cli history detail location** command at the EXEC mode as follows:

```
RP/0//CPU0:router# show cli history detail location 0/RP0/CPU0
Sun Jun 13 21:52:10.219 DST
 No.  Username        Line          Client    Time             Command
---- --------------- --------------- -------------------- ----------------------------
   1   lab      vty0       adminexec  Mon May 31 22:10:23.156 PST show configuration
commit list
   2   lab      vty0            adminexec  Mon May 31 22:10:31.352 PST exit
   3   lab      vty0                 exec  Mon May 31 22:10:45.627 PST admin
   4   lab      vty1                 exec  Mon May 31 22:12:03.853 PST configure
   5   lab      vty1               config  Mon May 31 22:12:06.463 PST mpls traffic-eng
```

The **detail** option displays the commands from each session along with user name and vty id so that commands issued from a session can be related with the session history displayed in the **brief** option.

Note The default size is 500 for the brief option of the command. The default size is 1000 for the detail option of the command.

Key Combinations

The following sections provide information on key combinations:

- Key Combinations to Move the Cursor, page 131
- Keystrokes to Control Capitalization, page 132
- Keystrokes to Delete CLI Entries, page 133
- Transposing Mistyped Characters, page 133

Key Combinations to Move the Cursor

Table 30 shows the key combinations or sequences you can use to move the cursor around on the command line to make corrections or changes. When you use cursor control keys, consider the following guidelines:

- Ctrl indicates the Control key, which must be pressed simultaneously with its associated letter key.
- Esc indicates the Escape key, which must be pressed first, followed by its associated letter key.
- Keys are not case sensitive.

Table 30 Key Combinations Used to Move the Cursor

Keystroke	Function	Moves the Cursor
Left Arrow or **Ctrl-B**	Back character	One character to the left. When you enter a command that extends beyond a single line, you can press the **Left Arrow** or **Ctrl-B** keys repeatedly to scroll back toward the system prompt and verify the beginning of the command entry, or you can press the **Ctrl-A** key combination.
Right Arrow or **Ctrl-F**	Forward character	One character to the right.
Esc, **b**	Back word	Back one word.
Esc, **f**	Forward word	Forward one word.
Ctrl-A	Beginning of line	To the beginning of the line.
Ctrl-E	End of line	To the end of the command line.

Keystrokes to Control Capitalization

Letters can be uppercase or lowercase using simple key sequences. Table 31 describes the keystroke combinations used to control capitalization.

Note Cisco IOS XR commands are generally case insensitive and typically all in lowercase.

Table 31 Keystrokes Used to Control Capitalization

Keystroke	Purpose
Esc, **c**	Makes the letter at the cursor uppercase.
Esc, **l**	Changes the word at the cursor to lowercase.
Esc, **u**	Makes letters from the cursor to the end of the word uppercase.

Keystrokes to Delete CLI Entries

Table 32 describes the keystrokes used to delete command-line entries.

Table 32 Keystrokes for Deleting Entries

Keystroke	Deletes
Delete or **Backspace**	Character to the left of the cursor.
Ctrl-D	Character at the cursor.
Ctrl-K	All characters from the cursor to the end of the command line.
Ctrl-U or **Ctrl-X**	All characters from the cursor to the beginning of the command line.
Ctrl-W	Word to the left of the cursor.
Esc, **d**	From the cursor to the end of the word.

Transposing Mistyped Characters

To transpose mistyped characters, use the **Ctrl-T** key combination.